Developer's Interface Guide for IA-64 Servers (DIG64)

Version 1.0
November 1999

DIG64 Promoters	
COMPAQ	Compaq Computer Corporation
DELL	Dell Computer Corporation
FUJITSU COMPUTERS SIEMENS	Fujitsu Siemens Computers
HEWLETT PACKARD	Hewlett-Packard
intel	Intel Corporation
IBM	IBM
NEC	NEC Corporation

DIG64 Contributing Members	
	**Adaptec**
	**American Megatrends**
	**Bull**
	**Interphase**
	**LSI Logic**
	**Mylex**
	**Novell**
	**Oracle**
	**Phoenix Technologies**
	**Qlogic**
	**SCO**
	**SUN Microsystems** Sun, Sun Microsystems, and Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the US and other countries.

Notice: Implementations developed using the information provided in this specification may infringe the patent rights of various parties including the Promoters and parties involved in the development of this specification. An agreement to grant a limited patent license is available under the terms of a DIG64 Adopters Agreement, a copy of which may be obtained from http://www.dig64.org.

Contents

Figures

Tables

Chapter 1 Introduction

This chapter introduces the audience, scope, and purpose of the *Developer's Interface Guide for IA-64 Servers* [DIG64]. The chapter also summarizes the content of the guide and defines guideline compliance.

REFERENCES

DIG64 uses acronyms to reference other documents. For example, the acronym DIG64 refers to the *Developer's Interface Guide for IA-64 Servers*. These acronyms are unique and may be enclosed in brackets [DIG64]. For more information about the document referenced, see the References section in this guide.

1.1 Audience For This Guide

This guide is for use by hardware and software system designers and architects. It assumes that readers are familiar with the referenced technologies.

1.2 Scope Of This Guide

Release 1 of the guide targets the first generation of IA-64 server products.

The guide defines a set of baseline IA-64 system building blocks and software interfaces (see

Figure 1-1). All building blocks and software interfaces are from industry standards, initiatives or 64-bit Intel® architecture (IA-64) specifications that are ready for use in product designs.

The guide targets IA-64 servers, operating system interfaces, and devices where interoperability across platforms is an important design criteria. This includes systems designed to support multiple operating systems (although not necessarily concurrently), operating systems designed to run on servers from different vendors, and devices designed to work with multiple servers and operating systems.

The scope of this guide is not limited by server size, scalability, system architecture, or target market.

The guide does not address platform implementation, physical packaging, form factors, or environmental design. Products designed for proprietary systems are not in the scope of this guide.

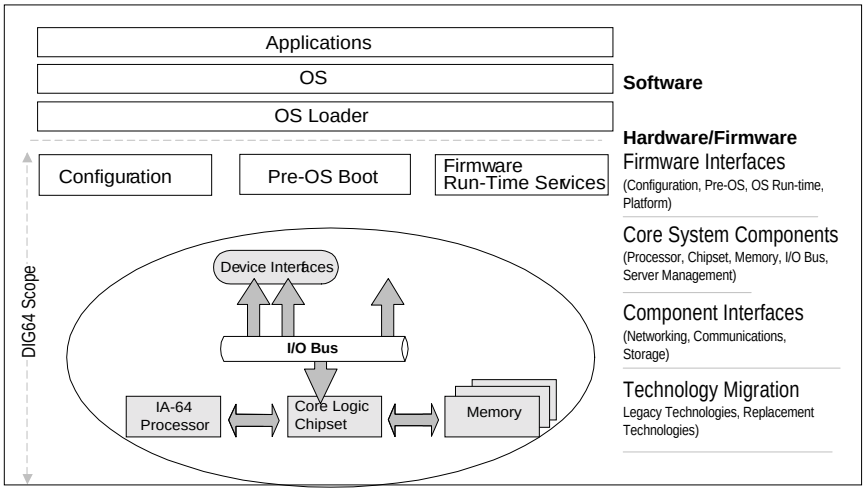

Figure 1-1: IA-64 Server Architecture and DIG64 Chapters

1.3 Purpose Of This Guide

The usefulness of any server product is based on the reliability and compatibility of the total system solution. Such solutions include IA-64-based servers, operating systems, add-in cards, and applications. When assembled in an operational system, the sum of these ingredients is a solution stack.

The *Developer's Interface Guide for IA-64 Servers* [DIG64] accelerates innovative, timely development of IA-64 solution stacks by removing obstacles to interoperability at the system level. DIG64 is an operating-system-independent reference guide that establishes basic system building blocks and defines required and optional interfaces between IA-64 based server hardware, firmware, and system-level software such as the operating system.

In addition to specifying interface guidelines, the guide documents industry consensus for migration from irrelevant legacy technology along with a roadmap for the introduction of replacement technologies. Legacy technologies, from older bus architectures to PC-AT boot procedures, impose a burden on system and OS designers and IT departments alike. This burden raises system and support costs and inhibits innovation and design flexibility without adding value to the end user.

The DIG64 guidelines provide industry consensus on the basic building blocks and software interfaces that form fundamental system designs and are intentionally set at a level where individual vendors can provide product differentiation. By providing this baseline consensus, the DIG64 is intended to produce the following benefits to adopters:

- **Optimize development resources** - By providing a baseline set of interfaces, the guide reduces the effort required by developers to negotiate interfaces with others in the industry and allows resources to be focussed on aspects of development.
- **Enable product innovation and differentiation** - The guide allows architects and developers to focus their effort on creative, innovative approaches to solving their customer's problems, instead of redefining basic interfaces.

- **Reduce time to market for IA-64 products** - By achieving industry consensus on the specified interfaces, the guide enables individual companies to accelerate design and development efforts, and accelerate the delivery of entire solutions by increasing the level of concurrent development that can take place at multiple levels of the solution stack.
- **Increase customer confidence and demand for products** - Developing products that adhere to standard interfaces improves reliability and availability. This provides lower qualification and support costs for IT customers and increased confidence in the solution stack.

1.4 Chapters Of This Guide

Chapters 1 and 2 provide summary information.

SUMMARY MODULE	CONTENT
Chapter 1: Introduction	This chapter introduces the audience, scope, and purpose of the *Developer's Interface Guide for IA-64 Servers* [DIG64]. The chapter also summarizes the content of the guide and defines guideline compliance.
Chapter 2: Guideline Summary	This chapter summarizes all guidelines. Guidelines are identified as required, recommended, or optional.

Chapters 3 through 8 address guidelines for base system technologies.

BASE SYSTEM MODULE	CONTENT
Chapter 3: Core Components	This chapter describes hardware and software structures used to build IA-64 systems.
Chapter 4: Firmware Services	This chapter names and positions specifications that define IA-64 firmware services.
Chapter 5: I/O Guidelines	This chapter addresses I/O subsystems.
Chapter 6: Server Management	The purpose of this chapter is to describe a set of features and functions in order to establish a baseline for management capabilities among IA-64 server systems.
Chapter 7: Technology Migration	This chapter addresses the migration of legacy hardware, firmware, and software.
Chapter 8: IA-32 Compatibility Considerations	This chapter provides optional guidelines for support of IA-32 Operating systems in IA-64 servers.

Chapters 9 and 10 address guidelines for key add-on components.

COMPONENTS MODULE	CONTENT
Chapter 9: Networking and Communications	This chapter seeks to specify the minimum guidelines for networking and communication peripheral attachment through the standard system bus.
Chapter 10: Storage Requirements	This chapter seeks to specify the minimum guidelines for storage peripheral attachment through the standard system bus.

1.5 Guideline Compliance

In order to ensure they receive the benefits of the guide, IT purchasers will look for DIG64

compliance as an attribute of IA-64 products. The sections below describe levels of guideline compliance.

Individual vendors are responsible for ensuring that their products meet the requirements of these guidelines before identifying a product as DIG64 Release 1.0 Compliant. DIG64 adopters may be provided tools, such as test suites, that will aid in determining adherence to the DIG64 requirements, as well as opportunities to test interoperability between products from multiple vendors.

1.5.1 What Is Required Compliance?

If a feature is *Required*, the feature must be implemented to comply with these guidelines. Other architectural layers assume the presence of a required feature.

NOTE

Some required features may be nested under parent features marked as recommended or optional. In this situation, the nested *Required* feature must be implemented for compliance only *if* implementers include the recommended or optional parent feature. These nested features are marked as *Required If* or *Recommended If*.

1.5.2 What Is Recommended Compliance?

If a feature is *Recommended*, the feature need not be implemented. Implementation is, however, encouraged. The presence of a recommended feature is not to be assumed nor precluded by other architectural levels.

An implementation which does not include a recommended feature must be prepared to operate with another implementation which does include the feature, though perhaps with reduced functionality. Similarly, an implementation which does include a recommended feature must be prepared to operate with another implementation which does not include the feature (except, of course, for the functionality the feature provides.) Recommended features may become requirements in the future.

1.5.3 What Is Optional Compliance?

If a feature is O*ptional*, the feature need not be implemented. There should be no dependencies on optional features by other architectural levels. Omitting an optional feature should have no impact on other required features or the proper functioning of the system.

Operating systems are not required to support optional features but must operate in the presence or absence of those features.

Chapter 2 Guideline Summary

The following tables provide a summary of the IA-64 guidelines for each chapter.

Section	Guideline Statement	Required, Recommended, or Optional
3.1.1	Core Chipsets Must Present Symmetric View Of System I/O And Memory	Required
3.1.2	Core Chipset Designs Allow For Identical Access Latencies	Recommended
3.1.2.1	Memory Systems That Support Non-uniform Memory Latencies Should Document The Systematic Asymmetric Latency Characteristics	Recommended, if NUMA architecture implemented
3.1.2.2	Memory Systems That Support Non-uniform Memory Latencies Should Provide System Performance Instrumentation	Recommended, if NUMA architecture implemented
3.1.3	Core Chipsets Must Permit Processor-To-Processor Interrupt Delivery	Required
3.1.4	Core Chipsets Should Maintain Symmetric View Of Interrupt Delivery	Recommended
3.1.5	Core Chipsets May Support Interrupt Delivery Based On Activity Level	Optional
3.1.6	Core Chip Set Internal Buses Should Have Error Detection Capabilities	Recommended
3.1.7	Internal Buses Should Have Sophisticated Error Tolerance Capabilities	Recommended
3.1.8	System Periodic Timer Must Be Provided	Required
3.1.8.1	Platforms That Support Multiple Periodic Tmers Should Provide A Means Of Synchronization	Recommended
3.1.9	Early Platform Boot Error Communication	Required
3.2.1	System Processors Must Execute The IA-64 Instruction Set	Required
3.2.2	System Processors Should Supply At Least One Low-Power State	Recommended
3.3.1	256 MB Memory Per Processor	Recommended
3.4.1	I/O Device Should Support APCI Power Management States	Recommended
3.4.2	The Hardware Implementation Of All Expansion Buses Must Provide A Means For Bus Agent Discovery And Identification	Required
3.4.3	Use Base Address Registers (BAR) And Extent Registers To Assign System Resources	Required
3.4.4	Dynamic Devices Should Not Use Fixed I/O Addresses	Recommended
3.4.5	Do Not Create Any New Fixed Address Functions	Required
3.5.1	Standby Power Should Be Provided To Enable ACPI Power Management Functions	Recommended

Section	Guideline Statement	Required, Recommended, or Optional
3.5.2	Power Subsystem Should Be Redundant	Recommended
3.5.2.1	Power Subsystem Should Provide Concurrent Serviceability	Recommended
3.6.1	EFI-compatible Firmware Must Be Used	Required
3.7.1	Nonvolatile Storage Should Be Provided For System Management	Recommended
3.7.2	Distribute Nonvolatile Storage Among Field Replaceable Units	Recommended
3.7.3	Standard Methods To Abstract Nonvolatile Storage Should Be Used	Recommended
3.8.1	ACPI Methods Should Be Used To Control Power Management And PnP-Related Hardware	Recommended
3.8.2	Provide ACPI Hardware For System Configuration	Required
3.8.3	ACPI Hardware For Power Management	Recommended
3.8.4	ACPI Registers Should Be Provided In The Memory-Mapped I/O Space	Recommended
3.8.5	ACPI Control Methods To Configure Devices Must Be Included	Required
3.8.6	Thermal Model And Fan Control	Recommended
3.8.7	ACPI Power State C1, C2, Or C3 Should Be Supported	Recommended
3.8.8	For Systems That Implement Power Management, ACPI-compliant Power Button Should Be Provided	Recommended
3.8.9	ACPI Real-Time Clock Should Be Provided	Recommended
3.8.10	Support ACPI Sleep States S0 And S5	Required
3.8.11	Support ACPI Sleep State	Recommended
3.8.12	USB Host Controller Should Be Able To Wake System	Recommended
3.9.1	SAPIC-compatible Programming Model Must Be Used	Required
3.10.1	Hardware And Firmware Must Support IA-64 MCA	Required
3.10.2	MCA Must Be Used For Error Reporting And Logging	Required
3.10.3	System Should Provide End-To-End Data Integrity	Recommended
3.10.4	No Breach Of Data Integrity Should Go Undetected	Recommended
3.10.5	Signal Corruption Of Data	Recommended
3.10.6	All Recovered Malfunctions Should Be Reported	Recommended
3.10.6.1	Error Reporting Should Not Impede Normal System Operation	Recommended
3.10.7	Detection And Correction Mechanisms Should Be Used	Recommended
3.10.8	Hot-Plug Technologies May Be Used	Optional
3.10.9	Deconfigure Failing Agents	Optional
4.1.1	IA-64 Firmware Must Implement SAL	Required
4.1.2	SAL Procedure Must Be Provided For Updating Firmware	Required
4.1.2.1	SAL May Provide Additional Authentication Features To Ensure That Only Approved Firmware Is Accepted	Optional

Section	Guideline Statement	Required, Recommended, or Optional
4.1.3	Software PCI Configuration Accesses Must Use SAL Procedures	Required
4.2.1	Machine Check Abort Support Must Be Implemented	Required
4.2.2	MCA Must Support Code Resources	Required
4.2.3	Nonvolatile Storage Must Be Provided For MCA Support	Required
4.2.4	Discontinue Use of PMI	Recommended
4.3.1	Extensible Firmware Interface Must Be Implemented	Required
4.3.2	Nonvolatile Storage Must Be Provided For EFI Support	Required
4.3.3	Serial Protocol Must Be Provided	Required
4.3.4	PXE Interfaces Should Be Provided	Recommended
4.4.1	Platform Configuration Using ACPI Methodologies Must Be Implemented	Required
4.4.2	Platform Power Management Should Be Implemented	Recommended
4.4.3	System Must Provide Multiple SAPIC Description Table	Required
4.4.4	Interrupt Assignments Must be Communicated through ACPI-compatible Namespace	Required
4.4.5	Provide A Differentiated System Description Table For Platform System Boards	Required
4.5.1	WfM Should Be Implemented	Recommended
4.6.1	Support For Updating Firmware Components Without Machine Reboot Should Be Provided	Recommended
4.6.2	Firmware Support For PCI Hot Plug	Recommended, if PCI Hot Plug is implemented
5.1.1	Implement I/O Bus	Required
5.1.2	The System Must Provide The I/O Subsystem With Access To All Of System Memory	Required
5.1.3	Adapters Must Support Transaction Initiation	Required
5.1.4	Systems Must Provide IO SAPIC Controllers For Interrupt Requests That Are Not Message Signaled	Required
5.2.1	Implement PCI Bus	Required, if PCI Bus is implemented
5.2.1.1	The PCI Bus And PCI Expansion Connectors Must Meet The Requirements Defined In PCI Local Bus Specification, Revision 2.1 Or Later	Required, if PCI is implemented
5.2.1.2	The PCI Bus And PCI Expansion Connectors Should Meet The Requirements Defined In PCI Local Bus Specification, Revision 2.2	Recommended, if PCI is implemented
5.2.1.3	Systems That Support More Than 4 GB Of Memory Must Include PCI Dual Address Cycle Capabilities For Inbound Transactions	Required, if PCI is implemented
5.2.1.4	Systems That Support More Than 4 GB Of Memory Should Include PCI Dual Address Cycle Capabilities For Outbound Transactions	Recommended, if PCI is implemented
5.2.1.5	PCI Bus Adapters and Embedded Devices Should Include Support For PCI Dual Address Cycles	Recommended, if PCI is implemented

Section	Guideline Statement	Required, Recommended, or Optional
5.2.1.6	PCI Devices Support Use Of Higher Level PCI Commands	Recommended, if PCI is implemented
5.2.1.7	Embedded Or Pluggable PCI Adapters Should Not Use The PCI Bus For Local Adapter Traffic	Recommended, if PCI is implemented
5.2.1.8	System Should Provide Sufficient Interrupt Request Lines	Recommended, if PCI is implemented
5.2.1.9	System Must Provide Support For Message Signaled Interrupts	Required, if PCI is implemented
5.2.1.10	Expansion Adapters And Devices Should Provide Support For Message Signaled Interrupts	Recommended, if PCI is implemented
5.2.1.11	System Must Not Contain Ghost Devices	Required, if PCI is implemented
5.2.1.12	PCI Devices Do Not Use The <1 MB BAR Type	Required, if PCI is implemented
5.2.1.13	System Must Use Standard Method To Close Base Address Register Windows On Nonsubtractive Decode PCI Bridges	Required, if PCI is implemented
5.2.1.14	VGA-Compatible Devices Must Not Use Nonvideo I/O Ports	Required, if PCI is implemented
5.2.1.15	System Must Support Configuration Of A Multi-Function PCI Device	Required, if PCI is implemented
5.2.1.16	Functions In Multifunction PCI Device Must Not Share Writable PCI Configuration Space Bits	Required, if PCI is implemented
5.2.1.17	Distinct Functions In Multifunction PCI Device Must Contain A Unique Device ID	Required, if PCI is implemented
5.2.1.18	Device Registers For Initialization, Configuration And Catastrophic Error Handling Must Be In The PCI Configuration Space	Required, if PCI is implemented
5.2.1.19	Device Registers For Normal Or Interrupt Operations Must Be Located In Memory-Mapped I/O Space	Required, if PCI is implemented
5.2.1.20	Device Registers For Normal Or Interrupt Operations May Be Located In I/O Port Space	Optional, if PCI is implemented
5.2.1.21	Device IDs Must Include PCI Subsystem IDs	Required, if PCI is implemented
5.2.1.22	PCI Devices Must Use Memory BARs	Required, if PCI is implemented
5.2.1.23	All PCI Agents Should Implement 3.3v Signaling	Recommended, if PCI is implemented
5.2.1.24	System Should Provide Support For 3.3 Vaux If System Supports S3 Or S4 State	Recommended, if PCI is implemented
5.2.1.25	PCI Peer-To-Peer Architectures Should Conform To A "Push-Only" Model	Recommended, if PCI is implemented
5.2.1.26	PCI Bus Subsystem Power Management	Recommended, if PCI is implemented
5.2.1.26.1	PCI Components Not Used On The System Board Must Comply With PCI Bus Power Management Interface Specification	Required, if PCI Subsystem power management is implemented
5.2.1.26.2	PCI Bus Power States	Required, if PCI Subsystem power management is implemented

Section	Guideline Statement	Required, Recommended, or Optional
5.2.1.26.3	CPU-To-PCI Bridge Power Management Must Be Implemented	Required, if PCI Subsystem power management is implemented
5.2.1.26.4	PCI-To-PCI Bridge Power Management Must Be Implemented	Required, if PCI Subsystem power management is implemented
5.2.1.27	CompactPCI* Specification May Be Used	Optional, if PCI is implemented
5.2.1.28	PCI Hot-Plug Specification Should Be Used	Recommended, if PCI is implemented
5.2.1.28.1	Systems That Support Hot-Plugging For Any PCI Device Should Use ACPI-Based Methods	Recommended, if PCI hot-plugging is implemented
5.2.1.28.2	Devices And Their Drivers Should Support ACPI State D3	Recommended, if PCI hot-plugging is implemented
5.2.1.29	Interrupt Binding on PCI Bus Adapters Must Follow the PCI-to-PCI Bridge Architecture Specification	Required, if PCI-to-PCI bridges are used
5.2.2	Implement PCI-X Bus	Required, if PCI-X bus is implemented
5.2.2.1	PCI-X Buses And Devices Meet Requirements For Device And Driver Support	Required, if PCI-X is implemented
5.2.2.2	PCI-X Devices Should Not Share Buses With Non-PCI-X Devices	Recommended, if PCI-X is implemented
5.3.1	Implement Universal Serial Bus	Optional
5.3.1.1	All USB Hardware Must Comply With USB 1.1 or Higher Specifications	Required, if USB is implemented
5.3.1.1.1	Controller Must Provide Full Bandwidth	Required, if USB is implemented
5.3.1.1.2	Controller Must be Located On I/O Expansion Bus	Required, if USB is implemented
5.3.1.2	USB Connections Must Use USB Icons	Required, if USB is implemented
5.3.1.3	USB Host Controller Must Comply With Compaq® OpenHCI Or UHCI Specification	Required, if USB is implemented
5.3.1.4	USB Devices And Drivers Must Not Use Isochronous Bandwidth For Alternate Setting 0	Required, if USB is implemented
5.3.1.5	USB Devices And Device Drivers Should Consume Bandwidth Only When They Are Being Used	Recommended, if USB is implemented
5.3.1.6	System And Devices Must Comply With USB Power Management Requirements	Required, if USB is implemented
5.3.1.7	USB Devices Must Comply With Their Related USB Device Class Specifications	Required, if USB is implemented
5.3.1.8	Bus-Powered USB Hubs Must Provide Ports That Can Be Individually Power-Switched	Required, if USB is implemented
5.4.1	I/O Subsystems May Implement I_2O	Optional
5.4.1.1	System Must Support A Messaging Interface	Required, if I_2O is implemented
5.4.1.2	An I_2O adapter Or Embedded Device Must Have The Class Code Programmed In The PCI Device Header	Required, if I_2O is implemented
5.4.1.3	An I_2O Adapter May Include Support For An Expansion ROM Interface	Optional, if I_2O is implemented
6.2.1	Event Logging	Required

Section	Guideline Statement	Required, Recommended, or Optional
6.2.2	Primary Management Channel	Required
6.2.2.1	Primary Management Channel Monitoring	Required
6.2.2.2	Primary Management Channel Alerting	Required
6.2.2.3	Primary Management Channel Event Log Access	Required
6.2.2.4	Primary Management Channel Event Log Storage	Required
6.2.3	Secondary Management Channel	Recommended
6.2.3.1	Secondary Management Channel Independent Of Host Processor	Recommended, if secondary management channel is implemented
6.2.3.2	Secondary Management Channel Independent Of Operating System	Recommended, if secondary management channel is implemented
6.2.3.3	Secondary Management Channel Independent Of Server Management Software	Recommended, if secondary management channel is implemented
6.2.3.4	Secondary Management Channel Independent Of System Power	Recommended, if secondary management channel is implemented
6.2.3.5	Secondary Management Channel Monitoring	Recommended, if secondary management channel is implemented
6.2.3.6	Secondary Management Channel Alerting	Recommended, if secondary management channel is implemented
6.2.3.7	Secondary Management Channel Event Log Access	Recommended, if secondary management channel is implemented
6.2.3.8	Secondary Management Channel Event Log Storage	Recommended, if secondary management channel is implemented
6.2.4	Platform Down Monitoring	Required
6.2.4.1	Platform Down Logging	Recommended
6.2.4.2	Platform Down Alerting	Recommended
6.2.4.3	Platform Down Recovery Action	Recommended
6.2.5	Server Physical Security Monitoring	Recommended
6.2.5.1	Server Physical Security Logging	Recommended
6.2.6	Cooling Device Failure Monitoring	Required
6.2.6.1	Cooling Device Failure Logging	Recommended
6.2.6.2	Cooling Device Failure Alerting	Recommended
6.2.7	Cooling Device Failure Predictive Indication	Recommended
6.2.7.1	Cooling Device Failure Predictive Indication Logging	Recommended
6.2.8	System Over-Temperature Monitoring	Required
6.2.8.1	System Over-Temperature Alerting	Recommended
6.2.8.2	System Over-Temperature Logging	Recommended
6.2.9	Processor Over-Temperature Monitoring	Required

Section	Guideline Statement	Required, Recommended, or Optional
6.2.9.1	Processor Over-Temperature Alerting	Recommended
6.2.9.2	Processor Over-Temperature Logging	Recommended
6.2.10	Processor Power Failure Monitoring	Required
6.2.10.1	Processor Power Failure Alerting	Recommended
6.2.10.2	Processor Power Failure Logging	Recommended
6.2.11	Power Supply Fault Monitoring	Required
6.2.11.1	Power Supply Fault Alerting	Recommended
6.2.11.2	Power Supply Fault Logging	Recommended
6.2.12	Power Supply Fault Predictive Indication	Recommended
6.2.12.1	Power Supply Fault Predictive Indication Logging	Recommended
6.2.13	System Power-Down Monitoring	Required
6.2.13.1	System Power-Down Logging	Recommended
6.2.14	System Reset Monitoring	Required
6.2.14.1	System Reset Logging	Recommended
7.3.1	ISA Expansion Slots Not Included Or Supported	Required
7.3.2	System Must Not Include Embedded ISA/LPC Network Adapters, Storage Controllers Or Graphics Adapters	Required
7.3.3	Support EFI Boot Loader For 64-bit OS	Required
7.3.4	Option ROM Support Using EFI	Recommended
7.3.5	Using EFI As The Pre-Boot Environment	Recommended
7.3.6	Architectural Support For Non-EFI Option ROMs	Optional
7.3.7	Architectural Support For DOS	Optional
7.3.8	Architectural Support For IA-32 Operating Systems	Optional
7.3.9	Architectural Support For Windows* 95 And Windows 98	Optional
7.3.10	Architectural Support For Serial COM Ports	Optional
7.3.11	Architectural Support For Parallel Ports	Optional
7.3.12	Architectural Support For PS/2 Ports	Optional
7.3.13	Architectural Support For USB Legacy Keyboard Emulation	Optional
7.3.14	Architectural Support For Floppy Disk Controller On SIO	Optional
7.3.15	Architectural Support For 8042 Keyboard Controller	Optional
7.3.16	Architectural Support For 8259A PIC	Optional
7.3.17	Architectural Support For 8254 Timer	Optional
7.3.18	Architectural Support For Real Time Clock Direct Accesses	Optional
7.3.19	Architectural Support For CMOS	Optional
7.3.20	Architectural Support For VGA	Optional
7.3.21	Architectural Support For Legacy Fixed Address Functions	Optional
8.1.1	Provide SAL Support For IA-32 Option ROMs and IA-32 Firmware Components	Optional

Section	Guideline Statement	Required, Recommended, or Optional
8.1.2	Provide IA-32 Support On Systems With EFI Interfaces	Optional
8.1.3	Provide Architectural Support For IA-32 Operating Systems	Optional
8.1.4	Provide Support For IA-32 Instruction Set Mode	Required, if support for IA-32 operating systems is provided
8.1.5	Firmware Support May Be Provided For IA-32 Emulation	Optional
9.1.1	Expansion Bus Specification Compliance	Required
9.1.2	Adapter Must Be Able To Transmit Packets From Buffers Aligned On Any Boundary	Required
9.1.3	Server Network Adapter Should Support Remote System Setup	Recommended
10.1.1	Expansion Bus Specification Compliance	Required
10.1.2	ANSI SCSI Compliance	Recommended
10.1.3	ANSI Fibre Channel Compliance	Recommended
10.1.4	ATA/ATAPI Compliance	Recommended
10.1.5	Option ROM Support	Optional

Chapter 3 Core Components

This chapter describes the hardware components and software structures used to build IA-64 server systems. It also names the standards that govern IA-64 server implementation.

Figure 3-1 is a conceptual drawing of an IA-64 server or node. It defines core system components as hardware, firmware, software elements, and the interfaces between them. Subsequent chapters cover core system component details. This chapter briefly touches on all components and highlights their interrelationships.

The drawing does not imply anything about partitioning or packaging. Blocks may consist of any number of physical components partitioned and packaged in various ways. Hardware components within each block may be placed on a single board, on multiple boards, or within multiple chassis. Servers may be interconnected to provide clustered or partitioned IA-64 systems.

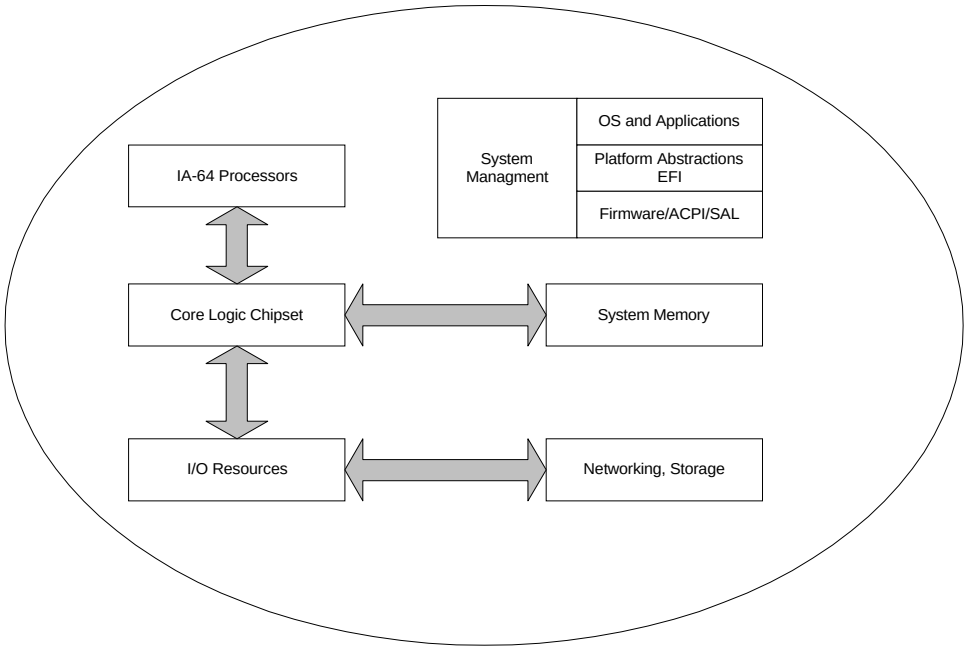

Figure 3-1: Conceptual Drawing of IA-64 Server

3.1 Core Chipsets

The core chipset is the central interconnection mechanism for the system. The chipset bridges the processor to memory and to I/O bus structures.

3.1.1 Core Chipsets Must Present Symmetric View Of System I/O And Memory

Required

The core chipset must present a symmetric view of system memory and I/O resources to all system processors. The term symmetric means that, after the system is fully configured by resident firmware, processors access memory and I/O resources in a programmatically identical way. For globally available resources, all system processors must see an identical system address map [RSDCA1, RSRASC1].

Memory may be accessed differently than I/O, meaning that different instructions may be used. Each processor, however, must see the same system programming model.

3.1.2 Core Chipset Designs Allow For Identical Access Latencies

Recommended

Access memory resources with identical latencies [RSDCA1, RSRASC1]. Also access I/O resources with identical latencies [RSDCA1, RSRASC1].

3.1.2.1 Memory Systems That Support Non-uniform Memory Latencies Should Document The Systematic Asymmetric Latency Characteristics

Recommended if NUMA architecture implemented

There are situations in which it may be an advantage to implement systems that present a Non-uniform Memory Access (NUMA) latency model to the system software. For these systems to achieve their potential performance the nature of the non-uniform access should be documented.

NOTE

At this time there is no standard or suggested method of describing resource affinities to an operating system.

3.1.2.2 Memory Systems That Support Non-uniform Memory Latencies Should Provide System Performance Instrumentation

Recommended if NUMA architecture implemented

Platform Providers should provide system instrumentation to enable characterization of workload-dependent latencies.

3.1.3 Core Chipsets Must Permit Processor-To-Processor Interrupt Delivery

Required

There must be a mechanism for a processor to send an interrupt message to any other system processor [RSPIA1, RSRASC1].

3.1.4 Core Chipsets Should Maintain Symmetric View Of Interrupt Delivery

Recommended

The core chipset may or may not be involved in the distribution of interrupts throughout the IA-64-based system. The system designer, however, should maintain the symmetric view with regard to interrupt delivery. Interrupt delivery from system I/O devices should be equally distributed among the available system processors [RSPIA1, RSRASC1].

3.1.5 Core Chipsets May Support Interrupt Delivery Based On Activity Level

Optional

The hardware may support delivery of interrupts based on the activity level of the system processors. The design should favor the processors doing the least amount of work when delivering interrupts [RSPIA1, RSRASC1].

3.1.6 Core Chip Set Internal Buses Should Have Error Detection Capabilities

Recommended

Internal buses within IA-64 servers use fast clock rates, wide data paths, and low-voltage electrical signaling. This creates a probability for transient errors. Buses should have error detection capabilities that minimize the possibility of data corruption when errors occur [IPMI1, RSEH1, RSSAL].

3.1.7 Internal Buses Should Have Sophisticated Error Tolerance Capabilities

Recommended

Internal buses (data, address, and control) should have error tolerance capabilities beyond that of simple error detection [IPMI1, RSEH1, RSSAL]. Elements of this functionality may include:

- Error correction on buses
- Hardware retry capability on detected errors
- Advanced detection schemes such as CRC

3.1.8 System Periodic Timer Must Be Provided

Required

The platform must provide a system periodic timer with interrupt capability. The implementation may consist of a single globally accessible timer, or of multiple distributed timers.


NOTE

Operating systems may use the IA-64 ITC for this purpose.

3.1.8.1 Platforms That Support Multiple Periodic Timers Should Provide A Means Of Synchronization

Recommended

In the case where multiple periodic timers are available, a means of precise synchronization should be provided. Synchronization may be achieved strictly by hardware or by a combination of hardware and software.

3.1.9 Early Platform Boot Error Communication

Required

IA-64 server platforms must provide a method to communicate to the user when errors are detected early in the boot sequence (prior to the ability to communicate those errors via the EFI console). Acceptable communication mechanisms can include but are not limited to flashing indicators or messages on integrated display panels. There is no requirement to provide access to this communication device to higher levels of the firmware/software if this mechanism is only used by the platform to communicate early boot errors. This replaces the use of the PC-AT speaker function of the 8254 timer, which the BIOS uses to communicate early boot failures by emitting beeps.

3.2 System Processors

3.2.1 System Processors Must Execute The IA-64 Instruction Set

Required

The system processors must execute the IA-64 instruction set architecture and possess all applicable programmatic interfaces, as documented in the applicable references [RSPPR1,

RSPPR2, RSPPR3, RSPPR4, RSSAL, RSRASC1].

Examples of programmatic interfaces include, but are not limited to:

- Virtual memory support
- Memory management unit organization
- Page table structures
- Translation registers
- Ordering models
- Semaphore support
- System register sets
- Interrupt models
- Firmware
- MCA

3.2.2 System Processors Should Supply At Least One Low-Power State

Recommended

System processors should support at least one low-power state. It should be possible for processors to enter or exit this state using a firmware call [RSPPR3].

Examples and guidelines covering the use of processor low-power states are present in [ACPI1].

3.3 System Memory

3.3.1 256 MB Memory Per Processor

Recommended

Each system should provide the capability of housing at least 256 MB of system memory per attached processor.

3.4 I/O Resources

The I/O resources of an IA-64 server consist of devices, device controllers, device drivers, and expansion buses. An example of an expansion bus is the Peripheral Component Interconnect and variants (PCI, PCI-X).

3.4.1 I/O Devices Should Support APCI Power Management States

Recommended

To facilitate ACPI compliant system low-power states, I/O devices should support ACPI low power states.

3.4.2 The Hardware Implementation Of All Expansion Buses Must Provide A Means For Bus Agent Discovery And Identification

Required

Provide a means for bus agent discovery and identification.

3.4.3 Use Base Address Registers (BAR) And Extent Registers To Assign System Resources

Required

The physical/virtual expansion bus address space visible to a system processor or SMP complex must be segmented by one or more BAR and extent registers. In general, any interconnect technology can be used as an expansion bus. For this requirement PCI, SCI, Synfinity, Myrinet, Servernet, and Memory Channel, to name a few, serve as examples of expansion bus technologies. With the exception of PCI, all others listed map an address of the local system processor to one or more global address spaces.

3.4.4 Dynamic Devices Should Not Use Fixed I/O Addresses

Recommended

Dynamic devices are defined as those devices that may come and go, either while the system is in use or across system boot sequences. Such devices should not use fixed I/O or memory address spaces. The design should provide a means for relocation in the system address map. For example, if the primary system bus is PCI, then PCI-defined mechanisms (BARs) should be used for device I/O and memory space assignment [PCI4].

3.4.5 Do Not Create Any New Fixed Address Functions

Required

As new functions are defined or as specifications for existing functions are updated, fixed I/O or memory address interfaces must be either refined or augmented to include relocatable interfaces.

The design should provide a means for relocation in the system address map. Use base registers (BARs) and extents for device I/O and memory space assignment.

3.5 System Power

3.5.1 Standby Power Should Be Provided To Enable ACPI Power Management Functions

Recommended

The system power subsystem should provide a standby power source to enable ACPI power management functions. A general guideline is that a standby power source should supply at least 5W of standby power, but this is platform-dependent. Voltages provided to the various standard buses (PCI, USB) must be in accordance with the relevant specification [ACPI1, PCI2].

3.5.2 Power Subsystem Should Be Redundant

Recommended

The power subsystem should be designed such that a portion of the subsystem can fail without interruption to system operation.

3.5.2.1 Power Subsystem Should Provide Concurrent Serviceability

Recommended If power subsystem is redundant

If the power subsystem supports redundancy, a failed power subsystem component should be replaceable without interruption of system power and/or system operation. Concurrent serviceability may be accomplished via hot-plug mechanisms, including add, replace and upgrade of a power subsystem without causing interruption to system availability and performance.

3.6 Platform Firmware Interface

3.6.1 EFI-compatible Firmware Must Be Used

Required

Extensible Firmware Interface (EFI) compatible firmware must be used for IA-64 servers. This ensures standardized APIs for booting multiple operating systems and for eliminating OS-specific loaders and legacy APIs [EFI1, IPMI1, RSSAL].

EFI abstracts system hardware from the operating system software and services.

3.7 Nonvolatile Storage

3.7.1 Nonvolatile Storage Should Be Provided For System Management

Recommended

IA-64 system designers should provide nonvolatile storage to enhance system management. This is in addition to other guidelines related to nonvolatile storage for preservation of important system data across boots.

3.7.2 Distribute Nonvolatile Storage Among Field Replaceable Units

Recommended

To enhance general system maintainability and to enable analysis of failed components, it is recommended that non-volatile storage be distributed to every extent possible across all system FRUs. The content and usage of the nonvolatile storage is vendor dependent. Examples include assembly information, error log data, VPD, and so on. This nonvolatile storage is apart from the nonvolatile storage used for preservation of important system data across system boots.

3.7.3 Standard Methods To Abstract Nonvolatile Storage Should Be Used

Recommended

Interfaces to nonvolatile storage should be abstracted through standard methods such as ACPI, IPMI, EFI, SAL, or MCA [ACPI1, EFI1, IPMI1, RSEH1, RSSAL].

3.8 Advanced Configuration And Power Interface (ACPI)

ACPI plays a major role in platform configuration and power management. For more information see Chapter 4 .

3.8.1 ACPI Methods Should Be Used To Control Power Management And PnP-Related Hardware

Recommended

For systems that implement power management and/or offer live insertion/removal of new hardware (hot-plug), ACPI methods should be used to implement the controlling hardware. Live insertion/removal means that the system is turned on and available for user processes.

3.8.2 Provide ACPI Hardware For System Configuration

Required

The following ACPI hardware must be provided to support ACPI system configuration:

- PM1a_EVT_BLK (PM1a_STS, PM1a_EN). Bit 0 (TMR) and 5 (GBL)
- PM1a_CNT_BLK (PM1a_CNT). Bit 0 (SCI_EN) and 2 (GBL_RLS)
- PM_TMR_BLK (PM_TMR). 32-bit timer is recommended
- SMI_CMD and the ACPI_ENABLE, ACPI_DISABLE value definitions
- GPE0_BLK (GPE0_STS and GPE0_EN). Only one bit is required
- Support for System Control Interrupt (SCI)

3.8.3 ACPI Hardware For Power Management

Recommended

Hardware as defined in the ACPI specification should be provided to support ACPI power management.

3.8.4 ACPI Registers Should Be Provided In The Memory-Mapped I/O Space

Recommended

ACPI registers should be provided in the memory-mapped I/O space [ACPI1, APCI2].

3.8.5 ACPI Control Methods To Configure Devices Must Be Included

Required

Each bus and device enumerated using ACPI must include the control methods necessary to configure the devices [ACPI1]. This requirement includes methods for automatic device configuration, resource allocation, and dynamic disable capabilities.

3.8.6 Thermal Model And Fan Control

Recommended

Thermal and fan control can be established in multiple mechanisms: OS presence, OS absence or both. If the implementation control policy needs OS participation, the control model must follow the ACPI definition [ACPI1].

3.8.7 ACPI Power State C1, C2, Or C3 Should Be Supported

Recommended

The system processor(s) should provide at least one ACPI-compliant reduced power state [ACPI1]. Use C1, C2, or C3.

3.8.8 For Systems That Implement Power Management, ACPI-compliant Power Button Should Be Provided

Recommended

Systems that have a power "button" and provide power-saving capabilities through software-controlled power management should implement this functionality in accordance with the ACPI specification [ACPI1]. The term "button" is defined as an actuation mechanism that can be used to change the system's power state.

3.8.9 ACPI Real-Time Clock Should Be Provided

Recommended

Provide an ACPI-compliant real-time clock alarm that supports wake-up based on time and day of the month. If this feature is implemented, the day-of-month feature is required. System control interrupt and STS/EN bits must also be provided [ACPI1].

3.8.10 Support ACPI Sleep States S0 And S5

Required

The following ACPI system sleep states must be supported: S0 and S5. This requirement does not mean that ACPI power management must be implemented, only that the sleep states S0 and S5 must be implemented per the definitions found in the ACPI specification [ACPI1].

3.8.11 Support ACPI Sleep State

Recommended

At least one of the following ACPI system sleep states should be supported: S1, S2, S3 or S4. This recommendation is in addition to the requirement to support S0 and S5; see section 3.8.10 .

This recommendation does not mean that ACPI power management should be implemented, only that if the sleep states S1, S2, S3 or S4 are implemented that they should conform to the definitions found in the ACPI specification [ACPI1].

3.8.12 USB Host Controller Should Be Able To Wake System

Recommended

If a system low-power state is supported and a USB host controller is present, the system should support wake-up capabilities from some or all of the system low-power state(s). If wake-up from the S2 or S3 state is supported, wake-up should be supported for all higher-power sleep states. For example, if the controller supports wake-up from the S2 state, it should also support wake-up from the S1 state [ACPI1].

3.9 Interrupt Programming Model

3.9.1 SAPIC-compatible Programming Model Must Be Used

Required

Regardless of the hardware interrupt delivery mechanism, interrupt controllers in IA-64 servers must use a programming model compatible with the APIC (Advanced Programmable Interrupt Controller) specification and its SAPIC extension for IA-64 processors [RSPIA1, RSRASC1, SAPIC1, SAPIC2]. See Chapter 4 for information.

3.10 Data Integrity

3.10.1 Hardware And Firmware Must Support IA-64 MCA

Required

Many features of IA-64 operating systems depend on platform support for IA-64 Machine Check Abort (MCA). Hardware and firmware must be implemented to support IA-64 MCA [IPMI1, RSEH1, RSSAL].

3.10.2 MCA Must Be Used For Error Reporting And Logging

Required

The IA-64 MCA has a complete method for reporting processor and many system errors. System

designers must use the MCA for system-wide error reporting and logging [RSEH1, RSSAL]. Use MCA to report platform errors, including errors within the system processors, memory, internal buses, and expansion buses.

NOTE

Certain established interfaces, such as TCP/IP in the networking space and SCSI in the storage space, have extensive error reporting and recovery mechanisms in their protocol stacks. Some interfaces (for example, serial ports, LPC, and so on) may have limited or no hardware support for reporting errors. Subsystems such as this are excluded from this requirement. Use of these interfaces should be avoided if an undetected error could lose or corrupt application data.

3.10.3 System Should Provide End-To-End Data Integrity

Recommended

It should be the goal of the system designer to guarantee the delivery of uncorrupted data to the consumer. This guideline impacts all major subsystems (processor, memory, and I/O) and signal pathways [ACPI1, IPMI1, RSEH1, RSSAL].

3.10.4 No Breach Of Data Integrity Should Go Undetected

Recommended

This requires active detection and notification circuitry for monitoring the address, control and data paths between subsystems.

3.10.5 Signal Corruption Of Data

Recommended

Any detected breach of data integrity should be signaled in a timely manner. This limits the possibility of delivering corrupted data to the consuming agent.

3.10.6 All Recovered Malfunctions Should Be Reported

Recommended

All recovered malfunctions or correctable data corruption should be reported to the platform.

3.10.6.1 Error Reporting Should Not Impede Normal System Operation

Recommended

The reporting of recovered malfunctions or correctable data corruption should not impede normal system operation.

3.10.7 Detection And Correction Mechanisms Should Be Used

Recommended

Standard single-bit correction and multibit detection mechanisms should be used [ACPI1, IPMI1, RSEH1, RSSAL].

3.10.8 Hot-Plug Technologies May Be Used

Optional

To facilitate repair, it may be desirable that FRUs be hot-pluggable [IPMI1, ACPI1, RSSAL].

3.10.9 Deconfigure Failing Agents

Optional

The system may have the ability to reboot itself after a fatal fault with the failing agent deconfigured [ACPI1, IPMI1, RSSAL].

Chapter 4 Firmware Services

This chapter names and positions specifications that define IA-64 firmware services. The figure below provides an overview of the components discussed.

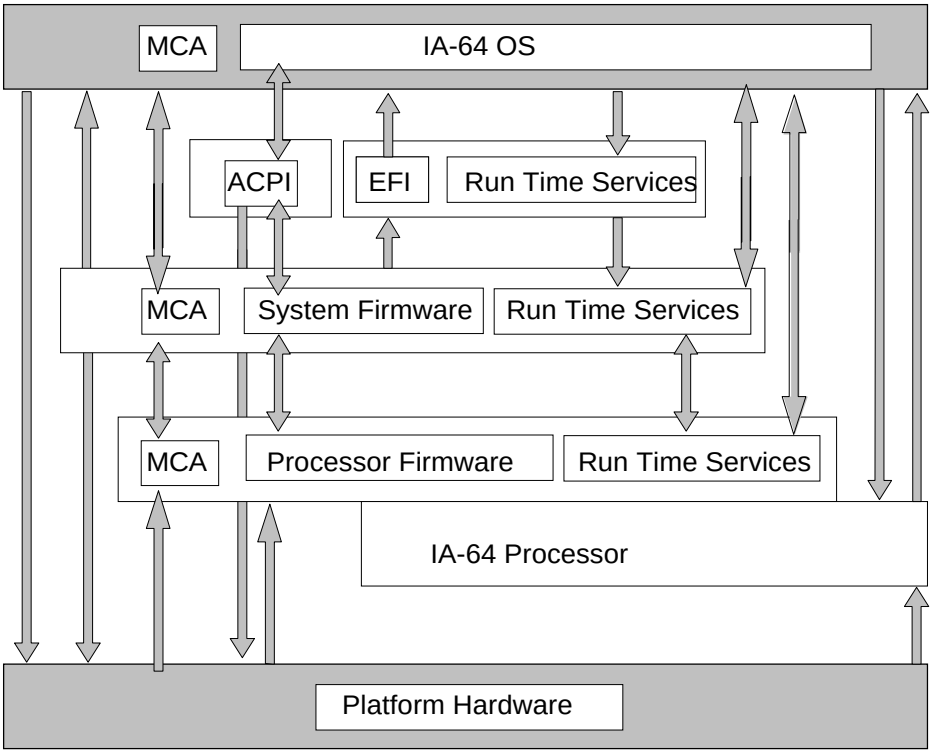

Figure 4-1: Firmware Components and Flow Control

4.1 System Abstraction Layer (SAL)

4.1.1 IA-64 Firmware Must Implement SAL

Required

The System Abstraction Layer (SAL) is a firmware layer provided by OEMs. The implementation of this layer must conform *to RS-IA-64 System Abstraction Layer (SAL) Specification*, Revision 2.6 or higher [RSSAL].

SAL abstracts platform uniqueness by providing a consistent interface to a higher level of the software stack to discover and control an IA-64 platform. It exports components and their associated access details to the OS through EFI using the SAL System Table.

4.1.2 SAL Procedure Must Be Provided For Updating Firmware

Required

The OEM must implement a call to update Intel provided firmware as specified in OEM/Intel firmware licensing.

4.1.2.1 SAL May Provide Additional Authentication Features To Ensure That Only Approved Firmware Is Accepted

Optional

The SAL authentication implies that the upgrade is validated and authorized by the system manufacturer.

4.1.3 Software PCI Configuration Accesses Must Use SAL Procedures

Required

In particular, access to PCI configuration space must not be made directly to the hardware.

4.2 IA-64 Machine Check Abort

4.2.1 Machine Check Abort Support Must Be Implemented

Required

IA-64 firmware must implement the IA-64 Machine Check Abort (MCA). Using MCA, every firmware layer in the system may check for recoverable errors, log errors, and perform error recovery where possible [RSEH1, RSSAL].

All machine check recovery code must use MCA firmware hooks to report platform errors.

4.2.2 MCA Must Support Code Resources

Required

All machine check recovery code must use MCA firmware hooks as specified in SAL 2.6 or higher [RSSAL] to report platform errors. Platform machine check hardware must provide relevant status bits for all sources of hardware errors that need to be handled in MCA code.

4.2.3 Nonvolatile Storage Must Be Provided For MCA Support

Required

This requirement is in addition to the nonvolatile storage needed and owned by all of the EFI-compliant operating systems that will boot on the platform.

4.2.4 Discontinue Use of PMI

Recommended

Use of PMI for reporting platform errors should be discontinued. This is in contrast to IA-32 platforms, where SMI (the IA-32 equivalent of PMI) may be used to implement machine check error logging and recovery.

On IA-64 platforms MCA provides greatly improved error recovery and logging capabilities, and should be used instead of PMI for reporting platform errors.

4.3 Extensible Firmware Interface Specification

4.3.1 Extensible Firmware Interface Must Be Implemented

Required

IA-64 firmware must implement the *Extensible Firmware Interface Specification* [EFI]. This specification describes the operating system (OS) interface to platform firmware. The interface defined consists of data tables that contain platform-related information. EFI also provides boot and runtime service calls that are available to the OS and its loader. Together, the specified tables and calls provide a standard environment for booting an OS.

4.3.2 Nonvolatile Storage Must Be Provided For EFI Support

Required

This is to support EFI environment variables.

4.3.3 Serial Protocol Must Be Provided

Required

IA-64 server systems must implement the serial protocol defined by [EFI1].

Server systems may use this protocol for debug applications and console redirection (headless

server implementations). The implementation of the protocol does not require the use of conventional COM port hardware. The code used to implement the protocol may use alternative devices (such as networking hardware) to support the capability.

4.3.4 PXE Interfaces Should Be Provided

Recommended

IA-64 server systems should implement the PXE protocol interfaces defined by [EFI1]. These interfaces support remote boot from the network.

PXE interfaces may also, with the addition TCP/IP support, present interactive tools that run in the preboot space for diagnostic, support, or configuration purposes.

4.4 Advanced Configuration And Power Management Interface Specification

4.4.1 Platform Configuration Using ACPI Methodologies Must Be Implemented

Required

IA-64 firmware must implement the Advanced Configuration and Power Management Interfaces (ACPI) defined by [ACPI1] or higher to configure platform resources.

4.4.2 Platform Power Management Should Be Implemented

Recommended

If IA-64 firmware implements power management, it must support the interfaces defined by [ACPI1] or higher to perform platform power management.

4.4.3 System Must Provide Multiple SAPIC Description Table

Required

The IA-64 server must include SAPIC support that complies with the IA-64 extensions to ACPI, implemented by providing the multiple SAPIC description table [ACPI2].

4.4.4 Interrupt Assignments Must be Communicated through ACPI-compatible Namespace

Required

The system must provide interrupt routing assignment information using a _PRT object as defined in the ACPI specification [ACPI1].

4.4.5 Provide A Differentiated System Description Table For Platform System Boards

Required

This table is defined in the ACPI specification [ACPI1, ACPI2].

4.5 Wired for Management Baseline Specification (WfM)

4.5.1 WfM Should Be Implemented

Recommended

IA-64 firmware should implement the *Wired for Management Baseline Specification* (WfM). This specification defines a baseline for the manageability that is used to lower the cost of computer ownership [WFMB1].

WfM requires the System Management BIOS (SMBIOS) table-based interface. This interface is used by the platform to relate platform-specific management information to the OS or to an OS-based management agent.

The format of SMBIOS table data is defined in [SMBR1]. It is up to higher-level software to map the data into the appropriate schema. Example schema include CIM (Common Information Model) and DMI (Desktop Management Interface).

The intent is to consider for adoption the various WfM specifications when they are 64-bit ready.

4.6 Dynamic Upgrades And Hot-plugging

4.6.1 Support For Updating Firmware Components Without Machine Reboot Should Be Provided

Recommended

Provide support for upgrading SAL and firmware components without requiring a machine reboot. The change may require a reboot to take effect.

4.6.2 Firmware Support For PCI Hot Plug

Recommended If PCI Hot Plug is implemented

If PCI Hot Plug is implemented, then the firmware should use ACPI methodologies to support it.

Chapter 5 I/O Attachment Guidelines

This chapter defines specific requirements for buses and devices provided in an IA-64 server.

Manufacturers who want to select high-performance components for servers should look for these design features in I/O components. Not all technologies described here are required for the IA-64 server. However, if the technology is implemented, it must conform as described herein.

5.1 Generic I/O Bus Requirements

This section summarizes requirements for the generic I/O bus, including the PCI bus. The three requirements in this section apply to all buses that may later become part of the approved server architecture.

5.1.1 Implement I/O Bus

Required

See the appropriate sections below for information regarding implementing a specific I/O expansion bus.

5.1.2 The System Must Provide The I/O Subsystem With Access To All Of System Memory

Required

The system must provide the hardware necessary for a device in the I/O subsystem to directly address all of the memory supported by the platform.

5.1.3 Adapters Must Support Transaction Initiation

Required

Adapters must support transaction initiation to provide more efficient use of the I/O bus.

5.1.4 Systems Must Provide IO SAPIC Controllers For Interrupt Requests That Are Not Message Signaled

Required

Systems must provide IO SAPIC Controllers as defined in [RSPIA1] for interrupt requests that are not message signaled. This includes the IO SAPIC programming model and the in-band interrupt delivery mechanism.

If a system only supports devices that are capable of generating message signaled interrupts, IO SAPIC controller is not needed.

5.2 PCI and PCI-X Bus Requirements

The guidelines defined in this section apply to a system that incorporates the PCI or PCI-X bus as an expansion bus. Most guidelines that reference PCI in this section apply equally to PCI-X. Note that guidelines discussed under Generic I/O Bus Requirements apply to the PCI or PCI-X bus as well.

5.2.1 Implement PCI Bus

Required If PCI Bus is implemented

If designers choose to implement the PCI bus, they must design to conform with the appropriate specification.

5.2.1.1 The PCI Bus And PCI Expansion Connectors Must Meet The Requirements Defined In PCI Local Bus Specification, Revision 2.1 Or Later

Required If PCI is implemented

The PCI bus and PCI expansion connectors must meet the requirements defined in [PCI1] or later.

5.2.1.2 The PCI Bus And PCI Expansion Connectors Should Meet The Requirements Defined In PCI Local Bus Specification, Revision 2.2

Recommended If PCI is implemented

If implementing the PCI bus, the PCI bus and PCI expansion connectors should meet the requirements defined in [PCI2].

5.2.1.3 Systems That Support More Than 4 GB Of Memory Must Include PCI Dual Address Cycle Capabilities For Inbound Transactions

Required If PCI is implemented

Systems that support more than 4 GB of memory must support the PCI Dual Address Cycle (DAC) for inbound accesses to pre-fetchable memory in the 64-bit physical address space. Since [PCI4] specifies that DAC cycles are to be used only for Memory Cycle Addressing at the 4 GB boundary or above, systems are still required to support the Single Address Cycle (SAC) for accesses below the 4 GB boundary.

NOTE

An Inbound transaction is initiated in the I/O subsystem by either an Embedded Device or a Bus Adapter. Completion of an Inbound transaction results in data being transferred between the System Memory and the Embedded Device or Bus Adapter.

5.2.1.4 Systems That Support More Than 4 GB Of Memory Should Include PCI Dual Address Cycle Capabilities For Outbound Transactions

Recommended If PCI is implemented

Systems that support more than 4 GB of memory should support the PCI Dual Address Cycle (DAC) for outbound accesses to pre-fetchable memory in the 64-bit physical address space. Since [PCI2] specifies that DAC cycles are to be used only for Memory Cycle Addressing at the 4 GB boundary or above, systems are still required to support the Single Address Cycle (SAC) for accesses below the 4 GB boundary.

NOTE

An Outbound transaction is initiated by the Host-to-I/O bus bridge. Completion of an Outbound transaction results in data being transferred between the System Memory or Host Processor and an Embedded Device or Bus Adapter.

5.2.1.5 PCI Bus Adapters and Embedded Devices Should Include Support For PCI Dual Address Cycles

Recommended If PCI is implemented

For maximum flexibility, PCI bus adapters and embedded devices should be able to access system memory over the entire 64-bit physical address space by providing support for Dual Address Cycles (DAC)s. Any pre-fetchable memory Base Address Registers should be 64 bits wide so that they can be mapped anywhere in the 64-bit address space. The Memory Base Address Register bits 2/1 should be encoded properly to indicate this as required by [PCI2].

5.2.1.6 PCI Devices Support Use Of Higher Level PCI Commands

Recommended If PCI is implemented

PCI devices should support use of higher level PCI commands, because they provide for more efficient prefetching and cache usage.

Higher level PCI commands are Memory Write and Invalidate (MWI), Memory Read Line (MRL), and Memory Read Multiple (MRM).

These higher level commands supplement the appropriate use of the Memory Write (MW) and Memory Read (MR) commands.

The higher level read commands (MRL and MRM) provide hints that can be utilized by the Host-to-PCI bridges to minimize the amount of prefetch overshoot when peripherals are requesting data from main memory.

The MWI command can be used by the Host-to-PCI bridges to invalidate entire cache lines when an upstream write is occurring, because MWI guarantees that the amount of burst data will be an integer multiple of the main memory's cache line size. Used in this way, the MWI command minimizes the overhead required of the Host-to-PCI bridge for large upstream memory write bursts, because the cache management activity only needs to occur on cache-line boundaries, instead of on the 8-byte (64-bit PCI bus) or 4-byte (32-bit PCI bus) boundaries that are necessary during a memory write.

5.2.1.7 Embedded Or Pluggable PCI Adapters Should Not Use The PCI Bus For Local Adapter Traffic

Recommended If PCI is implemented

To provide for more effective use of the PCI bus bandwidth and improve overall system performance, adapter cards should be designed so that they do not use the PCI bus to which they are attached for local adapter traffic.

5.2.1.8 System Should Provide Sufficient Interrupt Request Lines

Recommended If PCI is implemented

System should provide sufficient interrupt request lines to enable all PCI slots and PCI device types to obtain exclusive use of an interrupt line. This will minimize the number of interrupt sources the OS will need to poll when processing interrupt requests from the I/O subsystem.

5.2.1.9 System Must Provide Support For Message Signaled Interrupts

Required If PCI is implemented

As the I/O subsystems in servers become more complex, the requirement to provide each PCI slot and device access to a nonshared interrupt line becomes increasingly more difficult and expensive to implement on the system board. Thus, requiring the system to provide support for MSI as specified in [PCI2] will provide an infrastructure to help alleviate this burden.

It is expected that the physical (non-MSI) interrupt mechanism will be supported in the system, but that the MSI will be present to facilitate enhanced expandability.

5.2.1.10 Expansion Adapters And Devices Should Provide Support For Message Signaled Interrupts

Recommended If PCI is implemented

As the I/O subsystems in system designs become more complex, the requirement to provide each PCI slot and device access to a nonshared interrupt-line becomes increasingly more difficult and expensive to implement on the system board.

Therefore, it is recommended that expansion adapters and devices provide support for MSI as specified in [PCI4]. However, devices should continue to implement interrupt pins for backward compatibility with systems that do not support MSI. It is expected that the need for interrupt pins should diminish over time.

Devices that do not support interrupt pins due to pin constraints (rely on polling for device service) may implement messages to increase performance without adding additional pins.

Therefore, system configuration software must not assume that a message capable device has an interrupt pin.

5.2.1.11 System Must Not Contain Ghost Devices

Required If PCI is implemented

The system must not include any ghost devices, which are devices that do not correctly decode the Type 1/Type 0 indicator.

A PCI card should be visible through hardware configuration access at only one bus/device coordinate. This requirement does not preclude the existence of multi-function PCI devices.

5.2.1.12 PCI Devices Do Not Use The <1 MB BAR Type

Required If PCI is implemented

Devices, at a minimum, must take any 32-bit address.

5.2.1.13 System Must Use Standard Method To Close Base Address Register Windows On Nonsubtractive Decode PCI Bridges

Required If PCI is implemented

PCI-to-PCI bridges must comply with [PCI3]. Setting the Base Address Register to its maximum value and the limit register to zeros must effectively close the I/O or memory window references for that bridge Base Address Register.

5.2.1.14 VGA-Compatible Devices Must Not Use Nonvideo I/O Ports

Required If PCI is implemented

A VGA-compatible device must only use legacy I/O ports set aside for video devices in the PCI specification [PCI1].

5.2.1.15 System Must Support Configuration Of A Multi-Function PCI Device

Required If PCI is Implemented

Adapters may present themselves to the system as multi-function PCI devices. Thus, systems must support configuration of multi-function PCI devices in order to properly enable these adapters.

5.2.1.16 Functions In Multifunction PCI Device Must Not Share Writable PCI Configuration Space Bits

Required If PCI is implemented

The operating system must be able to treat each function of a multifunction PCI device as an independent device. There can be no sharing between functions of writeable PCI Configuration Space bits (for example, when the Command register for a multifunction PCI device is used to control multiple PCI functions).

This requirement is relaxed for logical devices within core chipset components that encompass more than one PCI function only to extend the address space. In this case, these logical devices must not share writable PCI configuration space bits with other independent devices.

5.2.1.17 Distinct Functions In Multifunction PCI Device Must Contain A Unique Device ID

Required If PCI is implemented

Operating systems treat each function in a multifunction PCI device as a separate device, therefore each distinct function in a multifunction PCI device must contain a unique device ID. Functions in a multifunction PCI device that are identical may contain the same device ID. Functions are deemed identical if they use the same driver interface.

5.2.1.18 Device Registers For Initialization, Configuration And Catastrophic Error Handling Must Be In The PCI Configuration Space

Required If PCI is implemented

Device registers for initialization, configuration and error handling must be in the PCI configuration space as part of the device dependent region (byte 64 to 255).

5.2.1.19 Device Registers For Normal Or Interrupt Operations Must Be Located In Memory-Mapped I/O Space

Required If PCI is implemented

This will allow more flexibility for the operating system to relocate the device control space than if I/O port space is used.

5.2.1.20 Device Registers For Normal Or Interrupt Operations May Be Located In I/O Port Space

Optional if PCI is implemented

Some legacy devices may need to be controlled through I/O port space to function properly. Using I/O port space ensures that the device control registers will be accessed using delayed transactions; thus a normal completion of a write will not occur until the register has actually been written. Posted transactions may occur through bridges in memory-mapped I/O space and will not guarantee that the register has actually been written when the initiating PCI master receives a normal completion.

5.2.1.21 Device IDs Must Include PCI Subsystem IDs

Required If PCI is implemented

The Subsystem ID (SID) and Subsystem Vendor ID (SVID) fields are defined in and required by [PCI2].

5.2.1.22 PCI Devices Must Use Memory BARs

Required If PCI is implemented

I/O Space is limited especially in hot plug systems and I/O references are generally slower than memory references. PCI devices are discouraged from using I/O Space and must use memory BARs. If the device uses I/O BARs (boot from IA-32 Option ROM being the reason), it must also use another BAR for memory references to the same set of registers.

5.2.1.23 All PCI Agents Should Implement 3.3v Signaling

Recommended If PCI is implemented

All PCI agents should implement 3.3v signaling as defined in [PCI2]. This may allow for removal of any 5-volt compatibility requirements for any future versions of this specification.

5.2.1.24 System Should Provide Support For 3.3 Vaux If System Supports S3 Or S4 State

Recommended If PCI is implemented

System support for delivery of 3.3Vaux to the PCI bus must be capable of powering a single PCI slot with 375 mA at 3.3V. System support must also be capable of powering each of the other PCI slots on the segment with 20 mA at 3.3V whenever the PCI bus is in the B3 state.

Systems must be capable of delivering 375 mA at 3.3V to all PCI slots whenever the PCI bus is in any "bus-powered" state: B0, B1, or B2.

5.2.1.25 PCI Peer-To-Peer Architectures Should Conform To A "Push-Only" Model

Recommended if PCI is implemented

Peer-to-peer architectures are used in PCI subsystem design when two adapters or embedded devices need to communicate directly with each other. There are two models by which this communication may be implemented, "push-only" and "push-pull". A "push-only" model allows exclusive use of memory writes for the peer-to-peer communication between two adapters or embedded devices. A "push-pull" model allows the use of both the memory write and memory read commands for peer-to-peer operation between two adapters or embedded devices. With the advent of multiple PCI segments in many I/O subsystems, there exists the possibility of not only intra-segment peer-to-peer transactions, but inter-segment peer-to-peer transactions. Inter-segment support of a "push-pull" peer-to-peer model has proven to be a very complex task for the host chipset, and as a result support for this has been inconsistent. Thus, PCI peer-to-peer architectures should conform to a "push-only" model.

5.2.1.26 PCI Bus Subsystem Power Management

Recommended if PCI is implemented

PCI bus sub-system power management should be supported. If PCI bus sub-system power management is supported, then the following guidelines must be implemented [PCI4].

5.2.1.26.1 PCI Components Not Used On The System Board Must Comply With PCI Bus Power Management Interface Specification

Required If PCI Subsystem power management is implemented

All expansion-capable devices on the PCI bus must comply with [PCI4]. This includes correct implementation of the PCI Configuration Space registers used by power management operations, and the appropriate device state (Dx) definitions for the PCI bus.

5.2.1.26.2 PCI Bus Power States

Required If PCI Bus Subsystem Power Management is implemented

The PCI bus must be in a bus state (B*x*) no higher than the system sleeping state (S*x*):

- If the system enters S1, the bus must be in B1, B2, or B3.
- If the system enters S2, the bus must be in B2 or B3.
- If the system enters S3, the bus must be in B3.
- In S4 and S5, system power is removed, so the bus state is B3.

A PCI bus segment must not transition to the B3 state until all downstream devices have transitioned to D3. Control of a PCI bus segment's power is managed using the originating bus bridge for that PCI bus segment.

5.2.1.26.3 CPU-To-PCI Bridge Power Management Must Be Implemented

Required if PCI Bus Subsystem Power Management is implemented

For CPU-to-PCI bridges, these controls must be implemented using [ACPI1] or [PCI4].

5.2.1.26.4 PCI-To-PCI Bridge Power Management Must Be Implemented

Required If PCI Bus Subsystem Power Management is implemented

For PCI-to-PCI bridges, these controls must be implemented in compliance [PCI4].

5.2.1.27 CompactPCI* Specification May Be Used

Optional If PCI is implemented

The CompactPCI* specification [CP1] offers an alternative mechanical form factor to that provided in the PCI bus specification. This mechanical implementation is acceptable for IA-64 systems if it is more appropriate for the target market of the system provider.

If used, CompactPCI card configuration mechanisms must fully comply with the [PCI2] specification.

5.2.1.28 PCI Hot-Plug Specification Should Be Used

Recommended If PCI is implemented

If implemented, one of two architectural specifications should be followed:

- PCI Hot-Plug Specification, Ver. 1.0 [PCI5]
- CompactPCI Specification [CP1]

The register-level implementation of a hot-plug controller is implementation-dependent. ACPI control methods, rather than device drivers, provide the system interface to these devices.

5.2.1.28.1 Systems That Support Hot-Plugging For Any PCI Device Should Use ACPI-Based Methods

Recommended If PCI hot plugging is implemented

The hardware insert/remove notification mechanism should be implemented as defined in [ACPI1].

5.2.1.28.2 Devices And Their Drivers Should Support ACPI State D3

Recommended If PCI hot plugging is implemented

To facilitate ACPI-compliant hot-plug implementations, devices and their drivers should support ACPI state D3.

5.2.1.29 Interrupt Binding on PCI Bus Adapters Must Follow the PCI-to-PCI Bridge Architecture Specification

Required If PCI-to-PCI bridges are used

The interrupt binding on PCI adapters must follow the PCI-to-PCI Bridge Architecture Specification [PCI3] for all PCI-to-PCI bridges, including nested bridges.

5.2.2 Implement PCI-X Bus

Required if PCI-X Bus is implemented

If designers choose to implement the PCI-X bus, they must design to conformance with the specification [PCI6].

5.2.2.1 PCI-X Buses And Devices Meet Requirements For Device And Driver Support

Required If PCI-X is implemented

Systems that provide PCI-X capabilities must meet the requirements defined by the PCI-X Addendum to the PCI Local Bus Specification Version 1.0 [PCI6] or later specifications plus other PCI-X and relevant PCI device driver requirements as defined by this guide.

5.2.2.2 PCI-X Devices Should Not Share Buses With Non-PCI-X Devices

Recommended If PCI-X is implemented

For the system to benefit from the performance enhancements of PCI-X, PCI devices should not be included on buses with PCI-X devices.

5.3 Universal Serial Bus (USB) Requirements

This section summarizes requirements for Universal Serial Bus (USB).

5.3.1 Implement Universal Serial Bus

Optional

Systems may implement USB.

5.3.1.1 All USB Hardware Must Comply With USB 1.1 or Higher Specifications

Required If USB is implemented

5.3.1.1.1 Controller Must Provide Full Bandwidth

Required If USB is implemented

The USB controller must provide full bandwidth.

5.3.1.1.2 Controller Must be Located On I/O Expansion Bus

Required If USB is implemented

The USB controller must be located on the I/O expansion bus.

5.3.1.2 USB Connections Must Use USB Icons

Required If USB is implemented

For USB connectors, the USB icon must be affixed to connectors and cables. The icon can be molded, printed, or affixed as a permanent sticker. Because the location and number of USB ports can vary, appropriate icons on both ports and cables are important for user ease-of-use. Therefore, USB icons are required for external cables, connecting cables, and connection ports.

Vendors can design their own icons, or they can use the recommended USB icon defined in [USB1].

5.3.1.3 USB Host Controller Must Comply With Compaq® OpenHCI Or UHCI Specification

Required If USB is implemented

The host controller must comply with the specifications for [OHCI1, UHC1]. Hardware manufacturers who design to one of these specifications are not required to provide an additional device driver for their host controller.

5.3.1.4 USB Devices And Drivers Must Not Use Isochronous Bandwidth For Alternate Setting 0

Required If USB is implemented

Device and driver designs should provide maximum flexibility of interface options to allow user-preference coordination by the operating system or other resource managers. This guideline helps enable flexible use of multiple simultaneous devices and applications in a dynamic environment.

5.3.1.5 USB Devices And Device Drivers Should Consume Bandwidth Only When They Are Being Used

Recommended If USB is implemented

Device and driver designs should provide maximum flexibility of interface options to allow user-preference coordination by the operating system or other resource managers. This guideline helps enable flexible use of multiple simultaneous devices and applications in a dynamic environment.

5.3.1.6 System And Devices Must Comply With USB Power Management Requirements

Required If USB is implemented

The system must comply with the power management requirements in [USB1]. USB devices must also comply with the Interface Power Management feature in [USB3].

5.3.1.7 USB Devices Must Comply With Their Related USB Device Class Specifications

Required if USB is implemented

A USB peripheral that fits into one of the USB device class definitions must comply with the related USB device class specification.

5.3.1.8 Bus-Powered USB Hubs Must Provide Ports That Can Be Individually Power-Switched

Required If USB is implemented

To minimize USB power consumption requirements, bus-powered hubs must provide ports that can be individually power-switched. This contributes to the goal of reducing overall system power.

5.4 Other I/O Architecture Issues

The requirements for new bus implementations will be included in future revisions of DIG64 as initiatives mature.

5.4.1 I/O Subsystems May Implement I_2O

Optional

Designers may choose to implement intelligent I/O. If so, it must meet the requirements defined in [I2O1].

5.4.1.1 System Must Support A Messaging Interface

Required If I_2O is Implemented

For designers implementing I_2O, an I_2O capable system must provide some sort of intelligence on the system board set or on an add-on adapter that enables sending and receiving messages, as defined in the I_2O specification.

5.4.1.2 An I_2O adapter Or Embedded Device Must Have The Class Code Programmed In The PCI Device Header

Required If I_2O is Implemented on PCI

For designers implementing I_2O, the PCI device header of an I_2O capable adapter or device must contain 0EH in the Class Code Register and 00H in the Sub Class Code Register. This indicates the device to be an I_2O intelligent device.

5.4.1.3 An I_2O Adapter May Include Support For An Expansion ROM Interface

Optional if I_2O is Implemented

This optional guideline supports functionality in systems with a standard firmware that isn't I_2O compliant.

Chapter 6 Server Management

6.1 Overview

The purpose of the server management chapter is to describe a set of features and functions in order to establish a baseline for management capabilities among IA-64 server systems. Typically, server systems are expected to be robust and resilient in the face of hardware, firmware and software malfunctions. Some server systems are designed with redundancy, fail-over and self-correcting characteristics that make such malfunctions invisible to the application. The aim of incorporating management functionality in IA-64 servers is to enhance the server's overall reliability, availability and serviceability.

Server management provides features and capabilities to monitor, predict and respond to problems that affect the overall health and functioning of the system. Server systems today employ a variety of management technologies and capabilities that are not consistent or standardized across the industry. For example, many server vendors have proprietary server management implementations that comprise that vendor's innovations. These proprietary implementations enable the vendors to differentiate their products from their competitors. Presently, there is no universal agreement among the major server vendors on a standard set of hardware and firmware interfaces to function with operating systems and enterprise management applications. Therefore, it is not possible at this time to document standard hardware guidelines that would suit every vendor's design requirements. Moreover, interoperability within the "server management stack" occurs at the top of the instrumentation layer (i.e. protocols and schema). See Figure 6-1 below. Note that the dotted line is meant to divide hardware and firmware components on the baseboard.

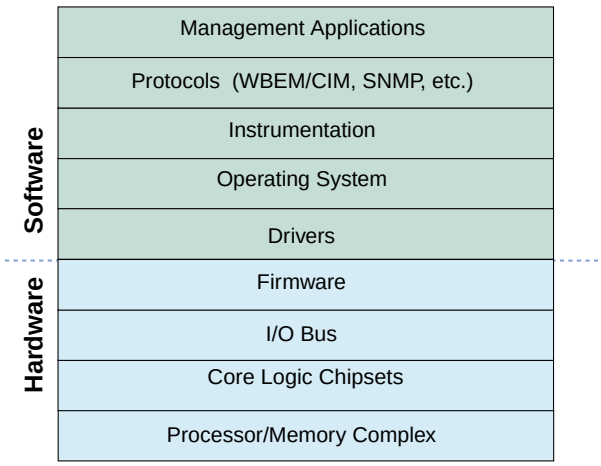

Figure 6-1: Server Management Stack

Specifying driver and instrumentation software interfaces is not within the scope of the DIG64 as described in Chapter 1 of this document. Note that the scope of the DIG64 is limited to platform hardware and firmware component interfaces. At the system board level, dedicated server management hardware such as service processors, emergency management ports, and other remote and out-of-band management hardware (i.e. independent of the operating system, the primary system bus and the central processing unit and core logic complex) may be implemented differently by each server vendor.

6.2 Baseline Guidelines

IA-64 server designers are encouraged to incorporate system management concepts and features in their architectures and systems designs. It is recommended that IA-64 servers provide the following set of hardware and firmware features to ensure that all systems will include the minimum baseline reliability, availability and serviceability capabilities. The guidelines discussed in this chapter are not an exhaustive enumeration of server management features and functions. Rather, they convey a set of the minimum capabilities that IA-64 servers should provide.

Also, the sets of alerts identified herein are examples of specific alerts commonly found on server systems and are not an exhaustive or functionally complete list. System vendors will most likely meet and exceed these items. There are no universal or common industry standards for which interfaces could be defined at the hardware level. However, within this context there is an acknowledgement that a certain level of server management functionality must be implemented to have a viable IA-64 server. Therefore, the minimum required features described herein must be supported even though they will be implemented in a variety of OEM-specific ways. System designers and developers must create instrumentation software that link these proprietary implementations to the operating system and to enterprise management applications through standardized management software, such as WBEM/CIM, SNMP [CIM1, SNMP1] and other relevant protocols, schema and specifications.

6.2.1 Event Logging

Required

The platform must provide the capability to log all critical events. It is required that an event log be made available for this purpose that can be accessed by management software.

6.2.2 Primary Management Channel

Required

The platform must provide a hardware interface that enables access to the system's local server management software or management agent. This can be via a standardized data transfer interface, such as 100baseT LAN, V.90 modem, fiber-channel, or proprietary hardware interfaces such as dedicated digital or analog signals, non-standard serial interfaces. To meet this requirement, the system is only required to provide a hardware interface that system management software can use for this purpose.

6.2.2.1 Primary Management Channel Monitoring

Required

The user must be able to monitor the server management functionality through the primary channel.

6.2.2.2 Primary Management Channel Alerting

Required

The platform must send alerts (i.e. make critical events data available to system management software) through the primary channel, if the primary channel is functional.

6.2.2.3 Primary Management Channel Event Log Access

Required

The platform must enable the primary management channel to access the system event log for recording critical events.

6.2.2.4 Primary Management Channel Event Log Storage

Required

The platform must provide event log storage that is accessible through the primary management channel.

6.2.3 Secondary Management Channel

Recommended

The platform should provide a hardware interface that enables access to the system's local server management software, event log, or agent when the primary management channel is unavailable. This can be via a standardized data transfer interface, such as 100baseT LAN, V.90 modem, fiber-channel, or proprietary hardware interfaces such as dedicated digital or analog signals, non-standard serial interfaces. To meet this recommendation, the system should provide a hardware interface that can be used to access management information under the described conditions.

6.2.3.1 Secondary Management Channel Independent Of Host Processor

Recommended if secondary management channel is implemented

The secondary management channel should operate independently of the host processor(s).

6.2.3.2 Secondary Management Channel Independent Of Operating System

Recommended if secondary management channel is implemented

The secondary management channel should operate independently of the operating system.

6.2.3.3 Secondary Management Channel Independent Of Server Management Software

Recommended if secondary management channel is implemented

The secondary management channels should operate independently of the server management software present on the system.

6.2.3.4 Secondary Management Channel Independent Of System Power

Recommended if secondary management channel is implemented

The secondary management channel should operate independently of the state of system power. For example, the channel should be available whether the system is on, off, or in a sleep state. A typical implementation may use an always-on standby power output from an AC-driven power supply to drive the secondary management channel and key elements of the platform management subsystem. A battery backup could cover conditions where AC power is not available.

6.2.3.5 Secondary Management Channel Monitoring

Recommended if secondary management channel is implemented

The user should be able to monitor the server management functionality through the secondary channel.

6.2.3.6 Secondary Management Channel Alerting

Recommended if secondary management channel is implemented

The platform should send alerts (i.e., make critical events data available to system management software) through the secondary channel, if the secondary channel is available. Typically, such an alert mechanism is independent of the installed operating system and is able to operate with minimal system functionality.

6.2.3.7 Secondary Management Channel Event Log Access

Recommended if secondary management channel is implemented

The platform should enable the secondary management channel to access the system event log for recording critical events.

6.2.3.8 Secondary Management Channel Event Log Storage

Recommended if secondary management channel is implemented

The platform should provide event log storage that is accessible through the secondary management channel.

6.2.4 Platform Down Monitoring

Required

Platform down detection is a hardware mechanism in the platform that supports the detection of

system software or processor hangs. A hardware watchdog timer is one way to implement this capability. For example, a common hardware watchdog implementation would allow system software to periodically access the timer in order to keep it from expiring. Upon expiration, the timer would typically provide a selectable set of actions, such as system power down, hard reset, power cycle, priority interrupt, and so on.

6.2.4.1 Platform Down Logging

Recommended

The occurrence of detected platform down events should be recorded in the system event log.

6.2.4.2 Platform Down Alerting

Recommended

The platform should produce an alert when the system is down or in a 'hung' state either due to hardware or software component failures.

6.2.4.3 Platform Down Recovery Action

Recommended

The platform should automatically trigger a recovery action in the event that the system is down or in a 'hung' state either due to hardware or software component failures (for example, power cycle, power down, reset, automatic hardware reconfigure, and so on).

6.2.5 Server Physical Security Monitoring

Recommended

The platform should be able to monitor the system's physical security (for example, whether any of the chassis doors have been opened). This type of monitoring is generally accomplished by switches connected to chassis doors and access panels.

6.2.5.1 Server Physical Security Logging

Recommended if server physical security monitoring is implemented

The occurrence of server physical security violations should be recorded in the system event log. For example, if a door or panel is opened, the event should be recorded.

6.2.6 Cooling Device Failure Monitoring

Required

The platform must detect the failure status of the system fans or other active cooling devices.

6.2.6.1 Cooling Device Failure Logging

Recommended

The failure events should be logged to the system event log.

6.2.6.2 Cooling Device Failure Alerting

Recommended

The platform should produce an alert when a cooling device failure event occurs.

6.2.7 Cooling Device Failure Predictive Indication

Recommended

The platform should monitor predictive (non-critical or warning) fan or active cooling device fault status, if such a feature is provided.

6.2.7.1 Cooling Device Failure Predictive Indication Logging

Recommended if cooling device failure predictive indication is implemented

Predictive fault events should be logged to the system event log.

6.2.8 System Over-Temperature Monitoring

Required

The platform must monitor for conditions that may lead to the system temperature exceeding a safe limit.

6.2.8.1 System Over-Temperature Alerting

Recommended

The platform should produce an alert when an internal system over-temperature event occurs.

6.2.8.2 System Over-Temperature Logging

Recommended

This event should be logged to the system event log.

6.2.9 Processor Over-Temperature Monitoring

Required

The platform must monitor for conditions that may lead to the processor temperature exceeding a safe limit.

6.2.9.1 Processor Over-Temperature Alerting

Recommended

The platform should produce an alert when a processor over-temperature event occurs.

6.2.9.2 Processor Over-Temperature Logging

Recommended

This event should be logged to the system event log, along with the identification of the processor for which the event has been detected.

6.2.10 Processor Power Failure Monitoring

Required

The platform must monitor for conditions that may lead to the failure of processor power.

6.2.10.1 Processor Power Failure Alerting

Recommended

The platform should produce an alert when the processor or processor module power source has failed.

6.2.10.2 Processor Power Failure Logging

Recommended

This event should be logged to the system event log.

6.2.11 Power Supply Fault Monitoring

Required

The platform must monitor the status of the primary power source. This includes redundant (i.e., distributed) power supplies.

6.2.11.1 Power Supply Fault Alerting

Recommended

The platform should produce an alert when a power supply fails.

6.2.11.2 Power Supply Fault Logging

Recommended

Power supply failures should be recorded in the system event log.

6.2.12 Power Supply Fault Predictive Indication

Recommended

If the power supply(s) supports it, the platform should be able to monitor predictive fault status from all power supplies, including redundant (for example, distributed) power supplies.

6.2.12.1 Power Supply Fault Predictive Indication Logging

Recommended if power supply fault predictive indication is implemented

Predictive power supply fault events should be logged to the system event log.

6.2.13 System Power-Down Monitoring

Required

The platform must detect the reason whenever the system powers down; for example, if the system power button was pressed or AC power to the system was lost.

6.2.13.1 System Power-Down Logging

Recommended

System power-down reasons should be recorded in the system event log. This information should be available after AC power is returned.

6.2.14 System Reset Monitoring

Required

The platform must detect the reason whenever the system is reset; for example, if the system-reset button was pressed or if software sent a reset command.

6.2.14.1 System Reset Logging

Recommended

System reset reasons should be recorded in the system event log.

Chapter 7 Technology Migration

This chapter proposes a path for the removal of legacy items from server platforms. The removal of an identified item is classified as *Required, Recommended,* or *Optional*.

Legacy migration is a scheduling exercise that is designed to keep suppliers of hardware, firmware, and operating systems in step.

7.1 Identifying Legacy Items

Legacy items are PC functions that are based on out-of-date technologies. Such items have been retained to ensure compatibility with older technologies. Legacy items are present in modern hardware, firmware, and operating systems.

In the server marketplace, especially in IA-64-based systems, legacy items frequently reduce the advantages of new technologies. Consider the following examples:

- Most servers are still configured with standard floppy disks. However, floppy disks no longer provide adequate backup or file transfer capability. Modern applications produce files that easily exceed disk storage capacity (graphics and other natural data types).
- Most servers have serial ports that once were used for modems. However, serial ports are now barely adequate for conventional modems. They do not support high-speed adapters.
- Most servers include parallel ports. Parallel ports once provided most print functionality. Now, however, printers are rarely found directly attached to servers.

The issue of legacy technologies in consumer PCs is being addressed in the industry by a broad-based initiative promoting greater ease of use in the platform. For more information see *Easier to Use Consumer PCs in 1999 Hardware Implementation Guide* [EASYPC].

7.2 Staging Legacy Removal

Legacy removal begins at Release 1.0 of this [DIG64]. The next formal release of this document following Release 1.0 defines a legacy-free server.

To support the migration away from legacy items, IA-64 platforms have legacy-free enablers added to the firmware and operating systems. These enablers are implementations of Extensible Firmware Interface (EFI) and other abstraction layers.

7.3 Legacy Technology Guidelines

This section details the guidelines to be followed to achieve a legacy-free server with the release of the IA-64 Developer's Interface Guide, following the Release 1.0. Note that omitting an optional feature must have no impact on other required features. The migration strategy for the

server and desktop industry may mutually influence each other. One industry may take the lead in areas where the restrictions of the legacy technology are more apparent. Table 7-1 summarizes all of the guidelines in this chapter. In the following table, the column "Future Release of the DIG64" represents a statement of direction of the DIG64 and not necessarily a commitment to the content or outcome of future releases of the DIG64.

Table 7-1: Legacy Guideline Summary

Legacy Technology Guidelines	Guideline Number	DIG64 R 1.0			Future release of the DIG64			Replacement Technology
		Req	Rec	Opt	Req	Rec	Opt	
ISA Expansion Slots not Included or Supported	7.3.1	✓			✓			PCI
System Must Not Include Embedded ISA/LPC Network Adapters, Storage Controllers, or Graphics Adapters	7.3.2	✓			✓			PCI
Support EFI Boot Loader for 64-bit OS	7.3.3	✓			✓			N/A
Option ROM Support Using EFI	7.3.4		✓		✓			EFI
Using EFI as the Pre-boot Environment	7.3.5		✓		✓			N/A
Architectural Support for Non-EFI Option ROMs	7.3.6			✓			✓	
Architectural Support for DOS	7.3.7			✓			✓	
Architectural Support for IA-32 Operating Systems	7.3.8			✓			✓	64 bit Operating Systems
Architectural Support for Windows* 95 and Windows 98	7.3.9			✓			✓	None
Architectural Support for Serial COM Ports	7.3.10			✓			✓	USB
Architectural Support for Parallel Ports	7.3.11			✓			✓	USB
Architectural Support for PS/2 Ports	7.3.12			✓			✓	USB
Architectural Support for USB Legacy Keyboard Emulation	7.3.13			✓			✓	
Architectural Support for Floppy Disk Controller on SIO	7.3.14			✓			✓	
Architectural Support for 8042 Keyboard Controller	7.3.15			✓			✓	USB
Architectural Support for 8259A PIC	7.3.16			✓			✓	SAPIC

Legacy Technology Guidelines	Guideline Number	DIG64 R 1.0			Future release of the DIG64			Replacement Technology
		Req	Rec	Opt	Req	Rec	Opt	
Architectural Support for 8254 Timer	7.3.17			✓			✓	
Architectural Support for Real Time Clock Direct Accesses	7.3.18			✓			✓	EFI
Architectural Support for CMOS	7.3.19			✓			✓	EFI
Architectural Support for VGA	7.3.20			✓			✓	EFI
Architectural Support for Legacy Fixed Address Functions	7.3.21			✓			✓	

7.3.1 ISA Expansion Slots Not Included Or Supported

Required

ISA slots are no longer required on servers. Functions formerly located on the ISA bus have migrated to other buses.

7.3.2 System Must Not Include Embedded ISA/LPC Network Adapters, Storage Controllers Or Graphics Adapters

Required

The benefits of an ISA-free system include improved performance, easier and more stable system configuration, and lower support costs.

7.3.3 Support EFI Boot Loader For 64-bit OS

Required

The 64-bit OS loader shall be implemented in accordance with the Extensible Firmware Interface (EFI) architecture specification [EFI1]. EFI abstracts the hardware interfaces and allows changing underlying code to permit substitution of new-for-old equivalent hardware. The loader is operating system neutral and provides a robust infrastructure for industry initiative support and an architectural means for OEM differentiation.

7.3.4 Option ROM Support Using EFI

Recommended

Option ROM support should follow the EFI specification [EFI1].

7.3.5 Using EFI As The Pre-Boot Environment

Recommended

DOS is currently the environment of choice in the pre-boot space. Manufacturing and test programs, some diagnostics programs, and many utility programs (including system management and set up routines for RAID controllers) use this pre-boot environment. Section 7.3.7 makes architectural support for DOS optional, so a replacement pre-boot environment must be identified. That replacement is EFI. EFI is available in the pre-boot environment and provides a set of services that can be used by test programs and utilities that used to depend upon DOS.

7.3.6 Architectural Support For Non-EFI Option ROMs

Optional

Support for non-EFI option ROMs may be provided.

7.3.7 Architectural Support For DOS

Optional

To support DOS, the system architecture must implement all the PC-AT legacy technologies (SIO, 8259A, 8042, and similar functions). Many of these technologies are planned to be migrated out of IA-64 servers. Making the support of DOS an optional guideline at Release 1.0 of the *Developer's Interface Guide for IA-64 Servers* allows OEMs, IHVs and ISVs the time to transition their utilities and test routines to an effective alternative. Supporting DOS in IA-64 servers is a hindrance to achieving legacy-free systems.

7.3.8 Architectural Support For IA-32 Operating Systems

Optional

Each OEM has the option to support any IA-32 operating system. Note that some of these operating systems will require legacy hardware and firmware to be present in the server platform to ensure correct operation.

7.3.9 Architectural Support For Windows* 95 And Windows 98

Optional

To support these operating systems, the server architecture must implement all the PC-AT legacy technologies (for example, SIO, 8259A, and 8042). Many of these technologies are planned to be migrated out of IA-64 servers. The support for the Windows* 95/98 operating systems in the IA-64 sever holds back the transitioning of legacy technologies and the related performance improvements.

7.3.10 Architectural Support For Serial COM Ports

Optional

PCs use COM ports to connect devices (e.g., serial printers and modems) and for computer-to-computer communication. COM port usage on servers is limited because of the low data rates supported. Other high data rate interfaces (LAN and USB) currently exist for devices that are traditionally attached to PC COM ports. Serial ports on servers have been used by operating system debug and server management functions mainly because of their simple interface.

7.3.11 Architectural Support For Parallel Ports

Optional

Support for parallel ports may be provided.

7.3.12 Architectural Support For PS/2 Ports

Optional

The OEM can retain the PS/2 ports for keyboard and mouse connection or use USB technology if a local console is implemented.

7.3.13 Architectural Support For USB Legacy Keyboard Emulation

Optional

USB legacy support requires SMI/PMI support and the 8042 keyboard controller to be present in the system.

7.3.14 Architectural Support For Floppy Disk Controller On SIO

Optional

For servers, low-capacity floppy drives are inefficient mechanisms for file backup and transfer. Alternative media exist that provide better capacity. Many OEMs have already moved to high-density floppy drives (such as the LS 120).

7.3.15 Architectural Support For 8042 Keyboard Controller

Optional

The 8042 keyboard controller function becomes optional on the IA-64 server if USB keyboard support and EFI (as the boot load process) are present in the system. Note that if the 8042 function is not implemented, DOS and some IA-32 based operating systems may not run on the server.

7.3.16 Architectural Support For 8259A PIC

Optional

For IA-64 systems, the SAPIC device replaces the 8259A PIC. The 8259A can be replaced or will be ignored by the firmware and OS since DOS is optional and EFI has replaced the older boot loader process. Depending on the particular IA-64 server design, the 8259A, if implemented, may be a separate chip or be part of the support chipset. In either case, the 8259A will be available for the OS and firmware but will be unused by a DIG64 compliant OS.

Note that if the 8259A function is not implemented, DOS and some IA-32 based operating systems may not run on the server.

7.3.17 Architectural Support For 8254 Timer

Optional

See Chapter 3 for information regarding timer support. The refresh request function is obsolete and does not have a replacement.

7.3.18 Architectural Support For Real Time Clock Direct Accesses

Optional

The EFI runtime services include a real time clock service. A direct hardware access mechanism to the real time clock (port 70h, 71h) is optional. Note that if such mechanism is not supported on a platform, DOS and some of the IA-32 based operating systems may not run on the server.

7.3.19 Architectural Support For CMOS

Optional

With EFI Variable Services for runtime, CMOS usage becomes optional. Note that if CMOS is not supported on a platform, DOS and some of the IA-32 based operating systems may not run on the server.

7.3.20 Architectural Support For VGA

Optional

With EFI console I/O protocol support at boot time, VGA dependency becomes optional. This will enable the headless support for IA-64 servers. Note that if VGA is not supported on a platform, DOS and some other operating systems may not run on the server.

7.3.21 Architectural Support For Legacy Fixed Address Functions

Optional

Support for legacy fixed address functions may be provided.

Chapter 8 IA-32 Compatibility Considerations

This chapter provides optional guidelines for support of IA-32 Operating systems in IA-64 servers.

8.1.1 Provide SAL Support For IA-32 Option ROMs and IA-32 Firmware Components

Optional

SAL mechanisms permit OEMs to support legacy IA-32 option ROMs and IA-32 firmware components. IA-32 code usage on an IA-64 platform is an OEM design choice [RSSAL].

8.1.2 Provide IA-32 Support On Systems With EFI Interfaces

Optional

EFI does not address a specific implementation strategy. The spectrum for EFI implementation ranges from legacy compatible to EFI only.

Server systems that support an IA-32 operating system in addition to an IA-64 operating system must implement legacy firmware and EFI interfaces [EFI1]. Support for an IA-32 operating system on an IA-64 server implies support for IA-32 boot interfaces and for IA-32 run-time accesses to firmware services.

NOTE

Support for existing PCI option ROMs (16-bit real mode) implies legacy firmware support. Refer to Chapter 7: Technology Migration. Such an implementation requires a full range of IA-32 services. EFI defines alternatives for option ROM replacement.

8.1.3 Provide Architectural Support For IA-32 Operating Systems

Optional

Each OEM has the option to support any IA-32 operating system. Note that some of these operating systems will require legacy hardware and firmware to be present in the server platform to ensure correct operation.

8.1.4 Provide Support For IA-32 Instruction Set Mode

Required if support for IA-32 operating systems is provided

While the processor is executing from the IA-32 instruction set within the IA-64 System Environment, the IA-32 application architecture as defined by the Pentium® processor must be used [RSPPR1].

8.1.5 Firmware Support May Be Provided For IA-32 Emulation

Optional

Firmware may contain IA-32 emulation code that, in conjunction with processor hardware, supports legacy IA-32 operating systems. This support is required if the target system must boot a legacy IA-32 OS [RSSAL, RSPPR1-RSPPR4].

Chapter 9 Networking And Communications

This chapter describes the requirements for network adapters. It is important to note that a network connection for the system, whether it is implemented on the motherboard or on an interface plugged into the system bus, must present a set of standard network connection services to any DIG-64 compliant operating system.

9.1 Network Adapter Requirements

9.1.1 Expansion Bus Specification Compliance

Required

Network adapters must be compliant with the specifications of the expansion bus to which they are attached.

9.1.2 Adapter Must Be Able To Transmit Packets From Buffers Aligned On Any Boundary

Required

Buffer alignment refers to whether a buffer begins on an odd-byte, word, double word, or other boundary. Adapters must be able to transmit packets, any of whose fragments are on an odd-byte boundary.

For performance reasons, packets should be received into contiguous buffers on a double-word boundary.

9.1.3 Server Network Adapter Should Support Remote System Setup

Recommended

It is recommended that server network adapters support remote new system setup capabilities as defined in [PXE2] or later.

Chapter 10 Storage Requirements

This chapter defines storage adapter requirements for IA-64 servers. It is intended for designers and integrators of compliant storage subsystems. See the *Storage Technical White Paper* [STWP] for more detailed information on storage technologies and possible implementations.

10.1 Adapter Requirements

10.1.1 Expansion Bus Specification Compliance

Required

Storage adapters must be compliant with the specifications of the expansion bus to which they are attached.

10.1.2 ANSI SCSI Compliance

Recommended

SCSI bus implementations should comply with ANSI specification X3.253 [SCSI1] created by ANSI X3T10 committee.

10.1.3 ANSI Fibre Channel Compliance

Recommended

Fibre Channel implementations should comply with ANSI specifications [FC1] through [FC4] created by ANSI X3T11 committee. Level of compliance varies with the level of implementation.

10.1.4 ATA/ATAPI Compliance

Recommended

If ATA/ATAPI is implemented, the interface must comply with the ANSI specification *Attachment with Packet Interface Extension (ATA/ATAPI-5)* [ATAPI5].

10.1.5 Option ROM Support

Optional

See Chapter 7 for guidelines on option ROM support.

References

A number of the references are Intel Confidential. These references are available through an Intel-approved source or an Intel representative.

ACPI1 *Advanced Configuration and Power Interface Specification, Revision 1.0b*
http://www.teleport.com/~acpi/

ACPI2 *RS-IA-64 Extensions to the Advanced Configuration and Power Interface Specification, Version 0.7 or later*

ATAPI1 *ATAPI Removable Media BIOS Specification (ARMD), Version 1.0*
http://www.phoenix.com/techs/specs.html

ATAPI4 *Attachment with Packet Interface Extension (ATA/ATAPI-4) NCTIS 317*
http://www.nssn.org

ATAPI5 *Attachment with Packet Interface (ATA/ATAPI-5)*
http://www.t13.org

CIM1 *Common Information Model (CIM) Specification, Revision 2.0*
http://www.dmtf.org/spec/cims.html

CP1 *CompactPCI Specification*
http://www.picmg.org/gabout.htm#Specifications

EFI1 *Extensible Firmware Interface Specification, Revision 0.91 or later*
http://developer.intel.com/technology/efi

EASYPC *Easier to Use Consumer PCs* in *1999 Hardware Implementation Guide*
http://developer.intel.com/technology/easeofuse/over.htm

FC1 *Fibre Channel - Physical and Signaling Interface (FC-PH) ANSI X3.230*
http://www.nssn.org/

FC2 *Fibre Channel - Physical and Signaling Interface (FC-PH) ANSI X3.297*
http://www.nssn.org/

FC3 *Fibre Channel - Physical and Signaling Interface (FC-PH) ANSI X3.303*
http://www.nssn.org/

FC4 *Fibre Channel Protocol (FCP) Rev 12 12-04-95 X3.269:1996*
http://www.nssn.org

IPMI1 *Intelligent Platform Management Interface (IPMI) Specification, Revision 1.0*
http://developer.intel.com/design/servers/ipmi/

OHCI1 *Open Host Controller Interface*
http://www.compaq.com/productinfo/development/

PCI1 *PCI Local Bus Specification, Revision 2.1*
http://www.pcisig.com/

PCI2 *PCI Local Bus Specification, Revision 2.2*
http://www.pcisig.com/

PCI3 *PCI-PCI Bridge Architecture Specification, Revision 1.1*
http://www.pcisig.com/

PCI4 *PCI Bus Power Management Interface Specification, Revision 1.1*
http://www.pcisig.com/

PCI5 *PCI Hot-Plug Specification, Revision 1.0*
http://www.pcisig.com/

PCI6 *Addendum 1.0 to PCI Local Bus Specification,*
http://www.pcisig.com/

PXE2 *Preboot Execution Environment (PXE) Specification, Version 2.0*
http://developer.intel.com/ial/wfm/wfmspecs.htm

RSDCA1 *RS-IA-64 System Design Considerations Application Note, Revision 1.0*
For information, contact your Intel representative.

RSEH1 *RS-IA-64 Error Handling Guide*
For information, contact your Intel representative.

RSPA1 *RS-IA-64 Platform Architecture Guide, Revision 1.5*
For information, contact your Intel representative

RSPIA1 *RS-IA-64 Processor Interrupt Architecture Guide, Revision 1.0*
For information, contact your Intel representative.

RSPPR1 *RS-IA-64 Processor Programmer's Reference Manual, Revision 3.0 or later, Volume 1: IA-64 Processor Application Architecture*
For information, contact your Intel representative.

RSPPR2 *RS-IA-64 Processor Programmer's Reference Manual, Revision 3.0 or later, Volume 2: IA-64 Processor System Architecture*
For information, contact your Intel representative.

RSPPR3 *RS-IA-64 Processor Programmer's Reference Manual, Revision 3.0 or later, Volume 3: IA-Instruction Set Description*
For information, contact your Intel representative.

RSPPR4 *RS-IA-64 Processor Programmer's Reference Manual, Revision 3.0 or later, Volume 4: IA-32 Instruction Set Description*
For information, contact your Intel representative.

RSRASC1 *RS-64-bit Runtime Architecture and Software Conventions for IA-64*
For information, contact your Intel representative.

RSSAL *RS - IA-64 System Abstraction Layer (SAL) Specification, Revision 2.6 or later*
For information, contact your Intel representative.

SAPIC1 *SAPIC EAS Streamlined APIC*
(this document is obsolete; see the reference [RSPIA1])

SAPIC2 *SAPIC Special Addendum*
(this document is obsolete; see the reference [RSPIA1])

SCSI1 *Small Computer System Interface-3 (SCSI-3)* ANSI X3-253
http://www.nssn.org/

SCSI2 *Small Computer System Interface-3 (SCSI-3)* ANSI X3-253, Appendix F
http://www.nssn.org/

SCSI3 *ANSI NCITS T10 SCSI-3 Multi-Media Command Set-2 (MMC-2)*
http://www.symbios.com/x3t10/drafts.htm

SDCPM1 *Storage Device Class Power Management Reference Specification, Revision 1.0*
http://www.microsoft.com/hwdev/onnow.htm#Specs

SMBR1 *System Management BIOS Reference Specification*
http://developer.intel.com/ial/WfM/design/BIBLIOG.HTM

STWP Storage Technology White Paper
http://dig64.org

SNMP *SNMP Introduction to Version 3 of the Internet-standard Network Management Framework*
http://www.ietf.org/rfc/rfc2570.txt

UHCI1 *Universal HCI (UHCI)*
http://www.intel.com/design/usb/designex/uhci11d.htm

USB1 *USB Specification*
http://www.usb.org/developers/

USB2 *USB Common Class Specification, Revision 1.0*
http://www.usb.org/developers/

USB3 *Universal Serial Bus Device Class Definition for Mass Storage Devices, Revision 1.0* http://www.usb.org/developers/

WFMB1 *Wired for Management (WfM) Baseline, Revision 2.0*
http://developer.intel.com/ial/wfm/

Glossary

ACPI (Advanced Configuration and Power Interface)
A specification that defines a new interface to the system board that enables the operating system to implement operating system-directed power management and system configuration.

CIM (Common Information Model)
A schema into which platform configuration data is mapped by higher-level software.

ECC (Error Correction Code)
A memory system that uses words with extra bits to expose memory errors. These extra bits are used as check bits in an error-correcting scheme that can detect and correct most memory errors.

EFI
Refers to the *Extensible Firmware Interface Specification*.

FRU
Field replaceable unit.

hot-plug
"Hot-plugging" refers to inserting new boards in a server that is already powered up and running. This can damage the new board or other parts of the machine unless the server is specially designed to allow insertion of new devices while the server is powered up.

IA-32
IA-32 is an abbreviation for Intel® 32-bit Architecture.

IA-64
IA-64 is an abbreviation for Intel® 64-bit Architecture.

legacy technology
A legacy technology is a computer subsystem that, although obsolete, is retained for compatibility. Support for legacy subsystems can minimize new technology benefits.

Inbound transaction
An Inbound transaction is initiated in the I/O subsystem by either an Embedded Device or a Bus Adapter. Completion of an Inbound transaction results in data being transferred between the System Memory and the Embedded Device or Bus Adapter.

Internal bus
A collection of signals internal to or between logical subsystems, which are confined within a chassis EMI enclosure, and whose malfunction may corrupt or lose application data.

ISA
Industry Standard Architecture.

LPC

Low Pin Count (LPC) bus.

MCA (Machine Check Abort)

The IA-64 Machine Check Abort is the system management and error correction approach built into the IA-64 processor family.

Outbound transaction

An Outbound transaction is initiated by the Host-to-I/O bus bridge. Completion of an Outbound transaction results in data being transferred between the System Memory or Host Processor and an Embedded Device or Bus Adapter.

PCI (Peripheral Component Interconnect)

A high-performance 32-bit or 64-bit bus designed to be used with devices that have high bandwidth requirements, such as the display subsystem.

PMI

Platform Management Interrupt

POST (Power On Self Test)

A set of diagnostic routines that is run each time a computer powers-on. These routines verify that the all hardware components—system board, memory, keyboard, etc.—are functioning correctly and initializes them.

SAL (System Abstraction Layer)

Firmware that abstracts system implementation differences.

SAPIC

Streamlined Advanced Programmable Interrupt Controller.

SIO

Super I/O controllers.

SMI

System Management Interrupt, the IA-32 equivalent of PMI.

SMBIOS (System Management BIOS)

A table-based interface that is required by WfM. It is used to relate platform-specific management information to the OS or to an OS-based management agent.

System Processor

A processor (CPU) that executes the IA-64 instruction set and is exposed to the global and symmetric system address space.

USB (Universal Serial Bus)

A bidirectional, isochronous, dynamically attachable serial interface for adding peripheral devices such as serial ports, parallel ports, and input devices on a single bus.

WfM (Wired for Management)

The *Wired for Management Baseline Specification* defines a baseline for system manageability issues. Its intent is to help lower the cost of computer ownership.

Index